ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської кваліфікаційної роботи

на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА WEB-ДОДАТКУ ДЛЯ ПОШУКУ ДРУЗІВ ТА ЗНАЙОМСТВ МОВОЮ JAVA З ВИКОРИСТАННЯМ SPRING FRAMEWORK**»

Виконав: студент 4 курсу, групи ПД– 44
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Левченко О.О.
(прізвище та ініціали)
Керівник Трінтіна Н.А.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)
Нормоконтроль
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Спеціальність - 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
_____О.В. Негоденко

«____»_____2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Левченко Олександр Олександрович
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка web-додатку для пошуку друзів та знайомств мовою Java з використанням Spring Framework»
Керівник роботи к.т.н., доцент Трінтіна Н.А.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «24» лютого 2023року №26.

2. Строк подання студентом роботи «1» червня 2023 року

3. Вихідні дані до роботи:
3.1.Офіційна документація Spring Framework
3.3.Офіційна документація IntelliJ IDEA від Jet Brains
3.4.Наукова-технічна література по розробці web-додатків

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1.Аналіз актуальності та огляд існуючих додатків
4.2. Аналіз інструментів реалізації web-додатк

4.3.Розробка структури та програмна реалізація.
4.4.Висновки

5. Перелік графічного матеріалу
5.1. Титульний слайд
5.2. Мета, Об’єкт та предмет дослідження
5.3. Завдання дипломної роботи
5.4. Таблиця порівнянь аналогів
5.5. Список головних функціональних та нефункціональних вимог
5.6. Програмні засоби реалізації
5.7. Діаграма варіантів використання
5.8. Діаграма класів
5.9. Екранні форми
5.10. Апробація результатів дослідження
5.11. Висновки

6. Дата видачі завдання «25» лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Утвердження теми бакалаврської роботи	25.02.2023	Виконано
2	Підбір науково-технічної літератури	26.02.2023 – 15.03.2023	Виконано
3	Аналіз актуальності та дослідження існуючих додатків	15.03.2023 – 20.03.2023	Виконано
4	Аналіз та вибір інструментів для розробки додатку	20.03.2023 – 14.04.2023	Виконано
5	Проектування та реалізація	14.04.2023 – 04.05.2023	Виконано
6	Вступ, реферат та висновки	04.05.2023 – 12.05.2023	Виконано
7	Попередній захист	15.05.2023 – 1.06.2023	
8	Захист роботи	12.06.2023	

Студент _____ Левченко О.О.
(підпис) (прізвище та ініціали)
Керівник роботи_____ Трінтіна Н.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Об'єкт дослідження – процес пошуку нових друзів та знайомств.

Предмет дослідження – спрощення процесу пошуку нових друзів та знайомств, через розробку web-додатку мовою Java з використанням Spring Framework та інших допоміжних інструментів для реалізації цього завдання.

Мета дослідження – допомогти полегшити пошук нових друзів та знайомств для людей за допомогою web-додатку.

У даній дипломній роботі досліджено позитивний вплив web-додатків для пошуку друзів та знайомств на підвищення рівня комунікації між людьми, через спрощення самого процесу пошуку людей для неї. Також досліджено актуальність даної теми, тобто такого типу web-додатків на Українському ринку. Було проведено порівняння з зарубіжними аналогами, названо їх переваги та вагомі недоліки.

Для створення додатку було використано й проаналізовано різні технології. А саме для реалізації було обрано Spring Framework через його популярність серед розробників, а також надійність і просту в розумінні документацію. Зі складу даного фреймворку було використано такі основні бібліотеки як Spring Security, Spring MVC.

Щодо БД , було досліджено інструменти для її створення, підключення та редагування:

- MySQL Workbench
- MySQL Server
- Hibernate & JPA

Щоб система мала високу надійність та продуктивність.

Архітектура ж додатку відповідає шаблону Model-View-Controller (MVC), для забезпечення поділу завдань і спрощення обслуговування коду.

За результатами проведених досліджень розроблено web-додаток який відповідає поставленій темі бакалаврської роботи.

ЗМІСТ

ВСТУП

Ідея даної досліджуваної теми бакалаврської роботи полягає у тому, щоб допомогти спростити процес пошуку нових знайомств та друзів через створення web-додатку. Опираючись на те, що на сьогоднішній день багато людей різних вікових категорій та статевих ознак покидають свої домівки та рідні міста. І через те що багато з них є самотні люди, ті хто не мають інших знайомих, або родичів в тих місцях куди вони переїжджають. Інші міста, особливо такі великі як Київ, Львів, або якісь менш відомі міста. Більшість таких людей надають перевагу залишатися вдома та не шукати комунікації з іншими людьми. Через це погіршується як і емоційний стан, так і фізичний стан.

Саме тому в даний час, web-додатки такого типу мають важливу роль в житті суспільства, адже вони саме створенні для того щоб полегшити людям пошук спілкування та комунікації з іншими.

Об’єктом дослідження даної дипломної роботи є процес пошуку нових друзів та знайомств.

Предмет дослідження представляє собою спрощення процесу пошуку нових друзів та знайомств за допомогою розробки web-додатку мовою Java з використанням Spring Framework та інших допоміжних інструментів для реалізації цього завдання.

Метою ж дослідження являється допомога полегшити пошук нових друзів та знайомств для людей за допомогою web-додатку.

Результати досліджень і розробки в роботі мають велике практичне значення для подальшого вдосконалення спрощення процесу пошуку друзів і нових знайомств. А також для навчання і майбутнього вдосконалення вмінь у розробці такого типу web-додатків за допомогою мови програмування Java та Spring Framework.

1.АНАЛІЗ АКТУАЛЬНОСТІ ТА ОГЛЯД ІСНУЮЧИХ ДОДАТКІВ

1.1 Популяризація пошуку нових знайомств через інтернет

В сучасному світі, з розвитком інтернет технологій. Спілкування в мережі є невід’ємною частиною людського життя.

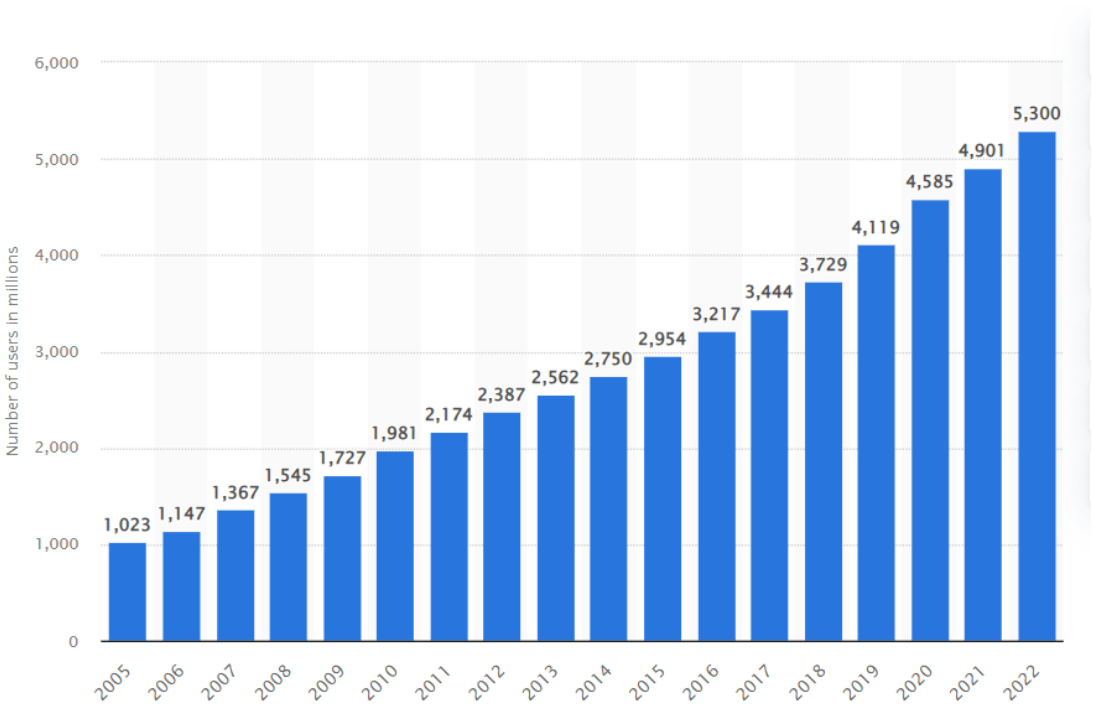


Рисунок 1.1 – Зростання кількості користувачів інтернету у світі

Це не дивно, адже віртуальний світ дозволяє знайти людей зі схожими інтересами та цінностями з будь-якої точки світу. Популяризація пошуку нових знайомств через Інтернет відкриває безліч можливостей для тих, хто хоче знайти нових друзів або навіть потенційних партнерів.

Один з найбільших переваг Інтернету полягає в тому, що ви можете знайти людей, які ділять ваші інтереси та цінності, з ким можна спілкуватися про все, що вас цікавить не виходячи з дому. Web-додатки для пошуку знайомств та друзів дозволяють вам створити профіль, де можна вказати ваші інтереси та інформацію про себе, які заходи ви полюбляєте, куди хотіли б сходити та інше. Це зробить вас більш привабливим для тих, з ким ви хочете спілкуватися.

Крім того, популяризація пошуку нових знайомств через Інтернет дозволяє знайти людей, яких ви не зустрінете у своєму повсякденному житті можливо через якісь труднощі, або просто якщо ви людина яка потребує поштовху. Саме Інтернет дає цей поштовх і можливість знайти нових знайомих з різних куточків країни чи навіть світу, які можуть поділитися з вами своїми думками та досвідом.

Однією з переваг популяризації пошуку нових знайомств через Інтернет є те, що ви можете комунікувати зі своїми новими друзями у будь-який час, зручним для вас способом. Більшість Інтернет-платформи для пошуку знайомств та друзів дозволяють вам використовувати чат для спілкування, що дозволяє знайти зручний спосіб для першого спілкування з вашими новими знайомими.

Однак важливо пам'ятати, що інтернет не є повноцінною заміною реального життя та особистих знайомств. Він може слугувати додатковим інструментом для знайомств і спілкування, але не може повністю замінити реальне спілкування.

1.2 Актуальність розробки web-додатків для пошуку друзів та знайомств

Завдяки розвитку нових технологій та додатків для пошуку друзів стало можливим швидко та ефективно знаходити співрозмовників з своєї місцевості.

Такі сервіси дають можливість знайти людей зі схожими інтересами та вподобаннями, які можуть стати добрими друзями або навіть потенційними партнерами для реальних зустрічей і подальших спільних подорожей. Спираючись на дослідження відомих американських психологів: П.Вацлавік, Д.Бівін та Д.Джексон у їхній роботі «Прагматика людських комунікацій» можна визначити особливості міжособистісного спілкування:

- Неуникність спілкування – кожен комунікує; для комунікації не існує свого антоніму «не-комунікації». Якщо особа не хоче контактувати, або спілкуватись з кимось. То не залежно від цього вона стане об’єктом обговорення для інших і проти власного бажання включається у загальну комунікацію.
- Неминучість спілкування – Інформація використана в повідомленнях

під час спілкування створює нові психологічні стани розуму і навіть нові становища тих, хто їх сприймає. Наслідки спілкування неминучі і це чітко підкреслено в прислів'ї: «Слово не горобець: вилетить – не спіймаєш».

- Двоаспектність спілкування – сенс його в тому, що учасники спілкування описують реальний світ, а також відповідні між ними відносини, такі як духовні, соціально-рольові, психологічні тощо. Цей аспект спілкування є реляційним, оскільки кожен вчинок щодо іншої особи містить інформацію, яка визначає і змінює відносини між ними.

Додатки для пошуку знайомств можуть використовувати алгоритми, які враховують географічне розташування користувача та його інтереси, щоб знайти найбільш підходящих людей для зустрічі. Це дозволяє користувачам знаходити тих, хто знаходиться поруч, забезпечуючи зручний і безпечний спосіб знайомства з новими людьми.

Особливо можуть бути корисними для людей, які переїжджають до нового міста чи країни і хочуть знайти нових друзів та підтримку в новому середовищі. Вони також можуть бути корисними для тих, хто не має достатньо часу, щоб знайти нових друзів офлайн.

Розглянемо декілька типів основної аудиторії користувачів, які можуть бути зацікавлені:

- Студенти та молодь, які покинули свої рідні міста та перейшли в інші вузи з тих чи інших причин, мають потребу у спілкуванні з ровесниками.
- Мешканці міста. Навіть якщо людина живе в одному місті все своє життя, вона може відчувати потребу у нових знайомствах та розвагах.
- Інші соціальні групи. Наприклад, люди з інвалідністю, які потребують підтримки спільноти людей з подібними потребами.
- Працівники які «релокейтнулись» тобто змінили місце роботи і переїхали в інше місто.

Проте основну увагу хотілося б звернути на таку групу людей як ВПО. Внутрішньо переміщені особи — це люди, які покинули свої домівки через

конфлікти, стихійні лиха чи інші небезпечні ситуації, але не перетнули кордон. Вони залишилися в межах своєї країни, проте втратили звичний спосіб життя та інфраструктуру.

Сьогодні через воєнні дії в нашій країні багато людей змушені покидати свої домівки. Основними причинами є близькість до лінії зіткнення, а також висока загроза життю. В Україні нараховується майже сім мільйонів внутрішньо переміщених осіб.

Рисунок 1.2- Інфографіка ВПО

Рисунок 1.3 – Кругова діаграма областей з яких частіше виїзжають ВПО

Для ВПО переїзд до інших містить носить великий стресовий характер. Особливо якщо там куди вони їдуть немає ні родичів, ні знайомих. В такій ситуації може виникнути почуття відчуженості та неприйняття, що може впливати на психологічний стан людини.

Щоб допомогти таким людям в адаптації до нового місця, важливо забезпечувати їм підтримку та можливості знайомитися з новими людьми. Саме тому розробка такого типу web-додатку може стати дуже корисним інструментом для цього.

Загалом, такі web-додатки для пошуку друзів і знайомств є важливими способами підтримки соціальної активності та популяризації здорового способу життя, що стає особливо потрібним у сучасному цифровому світі.

1.3 Огляд існуючих сервісів та засобів для пошуку друзів та знайомств

Історія створення сервісів для знайомств починається від початку самого розвитку інтернету та технологій. Перші сайти які з’явилися тоді, ще у 1990-х роках, якраз тоді коли інтернет почав тільки популяризуватися серед широкої аудиторії.

Одним із перших сайтів для знайомств можна назвати дітище компанії Match Group, сайт під назвою Match.com. Його безкоштовна бета версія була запущена 21 квітня 1995-го року, в цьому ж році його було представлено в відомому тоді журналі Wired. Цей журнал в ті роки публікував різні матеріали про комп’ютерні технології які мали широкий вплив, як на культуру так і економіку.

Рисунок 1.4 – Вигляд головної сторінки Match.com

Рисунок 1.5 – Зображення з телевізійна реклами сервісу

Останніми роками спостерігається значний зсув у бік використання онлайн-інструментів для пошуку друзів. З розвитком соціальних мереж і додатків для знайомств люди перестали обмежуватися особистими зустрічами або покладатися на знайомство з друзями та родичами.

Існує багато засобів які допомагають спростити пошук нових знайомств та друзів онлайн, за допомогою комп’ютерів та інших гаджетів. Неважливо чи у вас Android або IOS та інші.

Ось декілька прикладів:

- RentAFriend.com - найбільший і найпопулярніший сервіс оренди друзів у світі.
- Meetup: Meetup - це web-сайт і додаток, який дозволяє користувачам знаходити групи і події за інтересами.
- Bumble – мобільний додаток для пошуку знайомств, друзів і тд. для пошуку нових друзів є окрема функція під назвою Bumble BFF.
- Patook: Patook - це web-додаток, розроблений спеціально для пошуку платонічних друзів.

Щодо Українського ринку то можна назвати один із таких засобів:

- ZNAKOMKA – web-сервіс для пошуку друзів та знайомств;

1.3.1 RentAFriend.com

RentAFriend.com - це web-сервіс, який з'єднує людей, що шукають дружбу з тими, хто зацікавлений у тому, щоб стати другом за наймом. Сайт було запущено 2009 року, і відтоді він привернув до себе багато уваги, як позитивної, так і негативної.

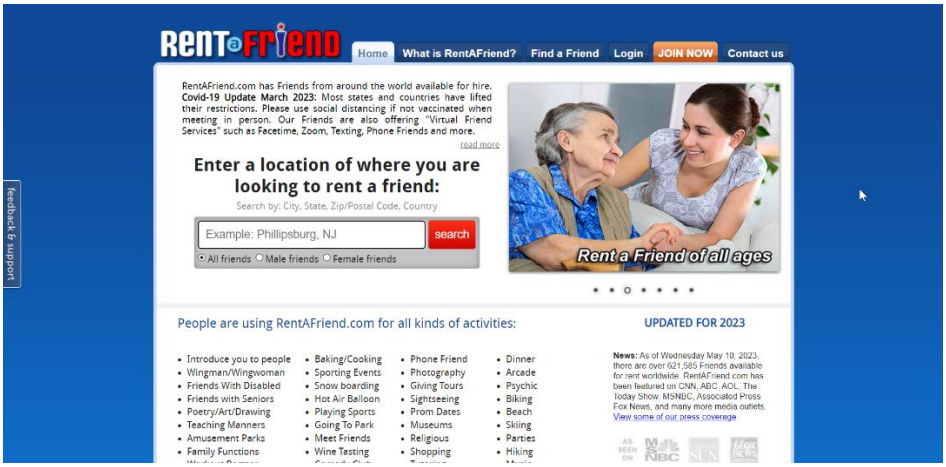


Рисунок 1.6 – Головна сторінка RentAFriend

Однією з головних переваг RentAFriend.com є те, що він пропонує альтернативний спосіб завести друзів, особливо для тих, хто бореться із соціальним

занепокоєнням або нещодавно переїхав на нове місце. Користувачі можуть шукати друзів на основі місця розташування, інтересів та інших критеріїв, а потім зв'язатися з ними, щоб домовитися про зустріч.

Ще однією перевагою RentAFriend.com є те, що він дає змогу користувачам заробляти гроші, будучи другом за наймом. Це унікальна концепція, яка виявилася успішною для сайту. Багато людей реєструються на сайті, щоб стати потенційними друзями. Оренда друга на кілька годин або на день може стати чудовим способом познайомитися з новими людьми і заробити трохи додаткових грошей.

Одні з основних переваги сервісу:

- Функція пошуку добре продумана і допомагає користувачам швидко знаходити потенційних друзів на основі їхнього місцезнаходження та інтересів.
- Велика база користувачів.
- Прибуткова модель, RentAFriend.com,стягує комісію з платежів.

Недоліки:

- Обмежена база користувачів у певних регіонах
- Відсутність вбудованої функції обміну повідомленнями
- Відсутність поглибленої верифікації та недостатність конфеденційності.

Загалом, RentAFriend.com - це цікава концепція, що має як позитивні, так і негативні сторони. Хоча він може бути не для всіх, він пропонує унікальний спосіб завести друзів і може бути цінним ресурсом для тих, хто хоче розширити своє коло спілкування.

1.3.2 Meetup

Meetup - це популярна соціальна мережева платформа, яка дає змогу людям створювати групи та організовувати заходи на основі спільних інтересів. Заснована 2002 року, платформа перетворилася на один із найбільших у світі web-сайтів для планування подій і соціальних мереж.

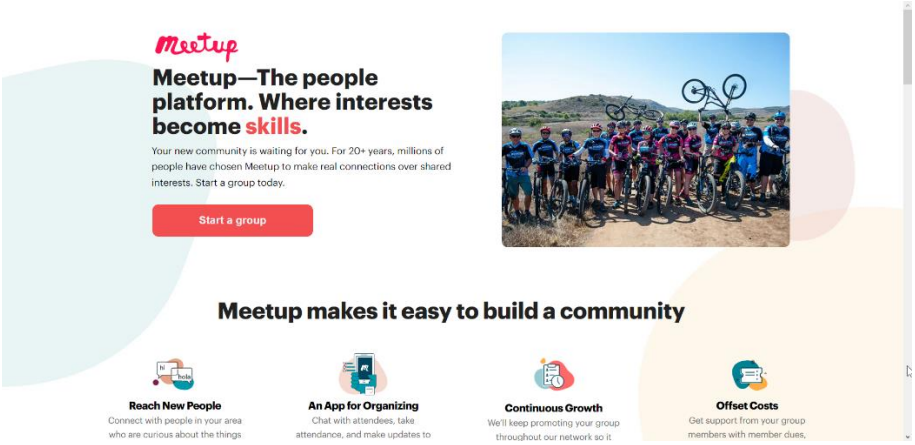


Рисунок 1.7 – Головна сторінка Meetup

Плюси:

- Платформа має велику базу користувачів, яка налічує понад 49 мільйонів членів у 193 країнах, що полегшує спілкування з однодумцями.
- Meetup має широкий спектр інтересів і тем, тож користувачі можуть знайти групи та заходи, які відповідають їхнім конкретним інтересам і хобі.
- Платформа є безкоштовною для приєднання та легкою у використанні, з простим користувацьким інтерфейсом та інтуїтивно зрозумілою навігацією.
- Meetup пропонує низку інструментів і функцій, які допомагають організаторам планувати й керувати заходами, наприклад, відстеження RSVP, обробка платежів та інструменти для комунікації.
- Meetup також пропонує мобільні додатки для iOS і Android, що дає змогу легко залишатися на зв'язку і бути в курсі подій і заходів.

Мінуси:

Хоча приєднання до Meetup є безкоштовним, деякі групи можуть стягувати плату за участь у заходах, що може бути перешкодою для деяких користувачів. Групи Meetup можуть мати обмежене географічне охоплення, тому користувачі з сільської місцевості або віддалених районів можуть мати менше можливостей для спілкування з іншими.

У минулому платформа піддавалася критиці за свою політику організації груп на спірні теми і була змушена закрити деякі групи через мовні ворожнечі та інші проблеми.

Бізнес-модель Meetup заснована на рекламі, що може призвести до захаращення головного вікна і відволікання користувачів.

З точки зору web-розробників, API та інструменти для розробників Meetup дають змогу легко створювати власні інтеграції та додатки на платформі. Однак API має деякі обмеження, і розробникам може знадобитися обійти ці обмеження для створення більш складних додатків. Крім того, зосередженість Meetup на рекламі може обмежити можливості деяких розробників монетизувати свої додатки.

Незважаючи на ці недоліки, Meetup залишається популярним вибором для користувачів, які шукають спілкування з іншими людьми та відвідують події, засновані на спільних інтересах.

1.3.3 Bumble і Bumble BFF

Bumble BFF - це функція популярного додатку для знайомств Bumble, яка допомагає користувачам знаходити платонічних друзів. Додаток дозволяє користувачам створювати профіль, а потім обирати потенційних друзів на основі їхніх інтересів, місцезнаходження та інших факторів. Якщо обидва користувачі проведуть по профілю один одного, вони можуть почати спілкуватися і, можливо, зустрітися особисто, щоб поспілкуватися.

Рисунок 1.8 – Інтерфейс Bumble BFF

Однією з переваг використання Bumble BFF є те, що він є розширенням великого додатку Bumble, який має добру репутацію у світі знайомств. Це означає, що додаток має велику базу користувачів і, як правило, добре розроблений і простий у використанні. Bumble BFF також має ряд функцій, які полегшують пошук спільних друзів, наприклад, можливість фільтрувати потенційні контакти за віком, інтересами та місцезнаходженням.

Недоліком використання Bumble BFF є те, що може бути важко відокремити його від основного додатку Bumble, який в першу чергу зосереджений на знайомствах. Це може призвести до плутанини серед користувачів, які шукають саме платонічну дружбу. Крім того, деякі користувачі повідомляли, що зустрічали в додатку людей, які шукали романтичних стосунків, а не дружби, що може розчаровувати тих, хто не зацікавлений у знайомствах.

Щодо виділення явних плюсів:

- Зосередженість на жіночій дружбі адже Bumble BFF був створений, щоб допомогти жінкам будувати дружбу в безпечному і сприятливому середовищі, що робить його чудовим варіантом для жінок, які шукають спілкування з іншими жінками.
- Верифікація профілю, Bumble BFF перевіряє профілі користувачів, щоб переконатися, що вони є легітимними, що може допомогти створити відчуття довіри та безпеки на платформі.
- Функція геолокації, Bumble BFF використовує геолокацію, щоб допомогти користувачам знайти потенційних друзів, які знаходяться поблизу, що полегшує особисту зустріч.
- Система обміну повідомленнями, система повідомлень на Bumble BFF проста у використанні і дозволяє користувачам швидко і легко спілкуватися з потенційними друзями.

Мінуси:

- Обмежена база користувачів. Bumble BFF не так широко використовується, як деякі інші платформи, що може обмежити кількість доступних потенційних друзів.
- Обмежена демографічна аудиторія, додаток в першу чергу орієнтований на молодих жінок, тому він може бути не настільки корисним для старших жінок або чоловіків, які шукають нових друзів.
- Обмежені можливості пошуку, що може ускладнити пошук потенційних друзів з певними інтересами або походженням.
- Покупки в додатку. Хоч і основні функції Bumble BFF є безкоштовними, є деякі додаткові функції, які вимагають оплати, що може відлякувати деяких користувачів.
- Bumble BFF має 24-годинний ліміт часу для відповіді на повідомлення, що може бути незручним для деяких користувачів, які не можуть відповісти в цей проміжок часу.

1.3.4 Patook

Patook - це соціальна мережа, яка має на меті об'єднати однодумців, які мають спільні інтереси та хобі, з акцентом на створенні платонічних дружніх стосунків. Додаток надає користувачам платформу для знайомства з новими людьми та спільної діяльності.

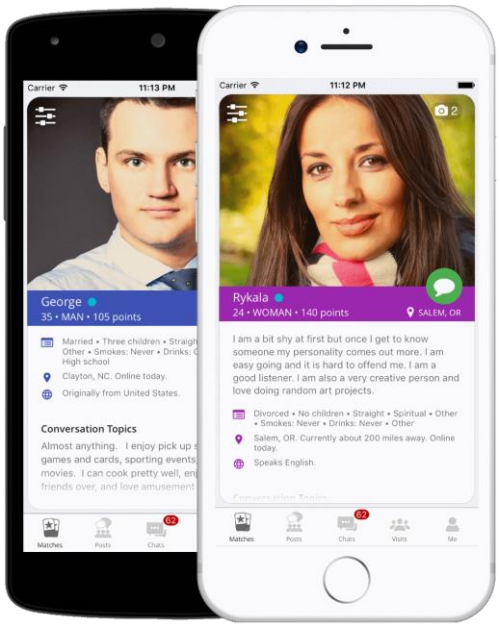


Рисунок 1.9 – Інтерфейс додатку Patook

Однією з головних переваг Patook є сувора політика проти флірту та побачень. Додаток призначений виключно для тих, хто шукає друзів і розвиває змістовні стосунки без жодного романтичного підтексту.

Patook також має унікальну "Систему балів", яка заохочує користувачів до позитивної взаємодії один з одним. Користувачі можуть заробляти або втрачати бали залежно від своєї поведінки, і ті, хто має багато балів, мають більше шансів бути представленими в додатку і рекомендованими іншим.

Однак деякі користувачі критикують Patook за його сувору політику і систему балів, які, на їхню думку, можуть бути занадто обмежувальними і створювати атмосферу конкуренції. Крім того, база користувачів додатку все ще відносно невелика порівняно з іншими соціальними мережами, що може ускладнити пошук відповідних збігів у деяких сферах.

Основні плюси додатку:

- Patook має унікальну функцію, яка дозволяє користувачам фільтрувати потенційних друзів на основі рівня їхнього інтересу до різних тем, що полегшує пошук однодумців.

- Додаток наголошує на створенні безпечного та позитивного середовища для пошуку друзів, із суворими рекомендаціями щодо неприйнятної поведінки та мови.
- Patook дозволяє користувачам створювати та приєднуватися до місцевих груп, заснованих на спільних інтересах, забезпечуючи більш цілеспрямований та орієнтований на громаду підхід до пошуку друзів.

Мінуси:

- Patook є відносно новим додатком і може мати не так багато користувачів у порівнянні з більш відомими додатками, такими як Bumble BFF або Meetup.
- Деякі користувачі повідомляли про проблеми з алгоритмом роботи додатку, що іноді може призвести до того, що збіги будуть не дуже сумісними.
- Додаток вимагає від користувачів платити за деякі функції, такі як розблокування можливості бачити, кому подобається ваш профіль.
- Акцент Patook на створенні позитивного і безпечного середовища може також призвести до того, що деякі користувачі відчувають, що додаток є занадто обмежувальним або не забезпечує достатньої свободи вираження поглядів.

1.3.5 Znakomka

ZNAKOMKA - популярний в Україні web-сервіс, який дозволяє користувачам шукати нових друзів, романтичних партнерів або просто співрозмовників для чату. Він був запущений у 2001 році і з тих пір став одним з найбільших сайтів знайомств і соціальних мереж в Україні.

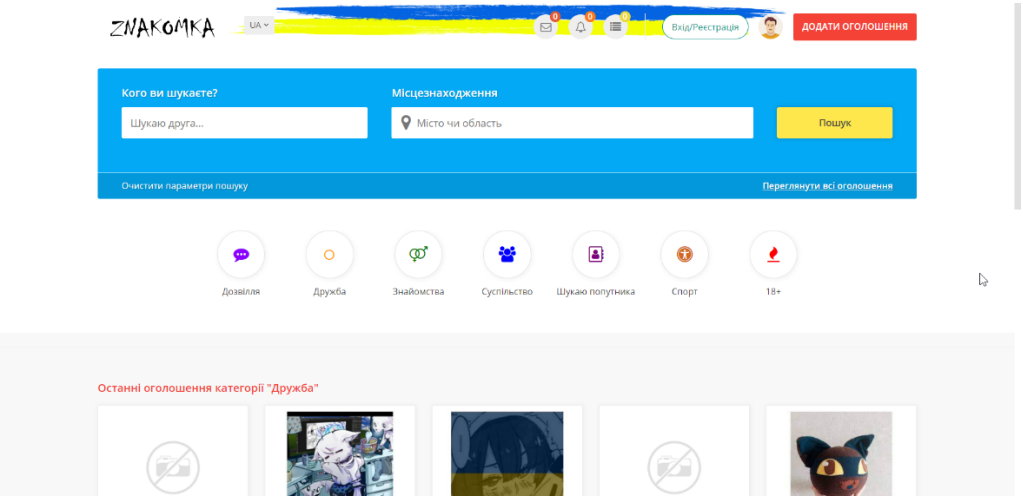


Рисунок 1.10 – Головна сторінка Znakomka

Однією з головних особливостей «Знакомки» є широкі можливості пошуку, які дозволяють користувачам фільтрувати потенційні контакти на основі широкого спектру критеріїв, включаючи вік, місцезнаходження, інтереси і так далі. Сайт також пропонує ряд комунікаційних інструментів, таких як чати та обмін повідомленнями, які допомагають користувачам спілкуватися один з одним.

На ZNAKOMKA є як безкоштовні, так і платні варіанти членства. Безкоштовні користувачі можуть створювати профіль, шукати інших користувачів, а також надсилати та отримувати повідомлення від інших користувачів. Платні користувачі мають доступ до додаткових функцій, таких як можливість переглядати фотографії інших користувачів і бачити, хто переглядав їхні профілі.

Як і у випадку з будь-яким сайтом знайомств або соціальною мережею, у використанні «Знакомки» є свої плюси і мінуси. До переваг можна віднести велику базу користувачів, широкі можливості пошуку і цілий ряд інструментів для спілкування. Однак деякі користувачі скаржаться на інтерфейс сайту і наявність фейкових профілів.

Загалом, ZNAKOMKA може бути корисним інструментом для тих, хто шукає нових знайомств, але важливо бути обережним і усвідомлювати потенційні ризики, пов'язані з онлайн-взаємодією.

Плюси:

- Досить відома та популярна платформа в Україні

- Безкоштовна у використанні, з можливістю оновлення до преміум-функцій за окрему плату
- Різноманітні функції, включаючи фільтри пошуку, обмін повідомленнями та чат
- Користувачі можуть вказати, чи шукають вони дружбу, романтичні стосунки або ділові контакти.
- Система верифікації для зменшення кількості фейкових профілів

Мінуси:

- Обмежена міжнародна база користувачів, в першу чергу орієнтована на Україну.
- Повідомлення про фейкові профілі або шахраїв на платформі
- Деякі користувачі повідомляли про агресивну або неналежну поведінку з боку інших користувачів, що може викликати занепокоєння щодо безпеки та комфорту на платформі

Таблиця 1.1 – Зведені результати аналізу характеристик сервісів та додатків для пошуку друзів та знайомств.

Назва	**RentAFriend**	**MeetUp**	**Bumble BFF**	**Patook**	**Znakomka**
Платформи	WEB	WEB	Android, IOS, WEB	Android, IOS	WEB
Безкоштовний доступ	+	+	+	+	+
Донат	+	+	+	+	+
Реклама	+	+	-	+	+
Платні підписки	-	+	+(Висока нав’язливість)	Частково	+
Заробіток для юзерів	+	-	-	-	-
Геолокація	-	+	+	+	+
Вбудований чат	-	+	+	+	+

Продовження таблиці 1.1 – Зведені результати аналізу характеристик сервісів та додатків для пошуку друзів та знайомств.

Назва	RentAFriend	MeetUp	Bumble BFF	Patook	Znakomka
Фільтри пошуку	+	+	+	+	+
Створення спільнот	-	+	-	-	-
Стабільність роботи	+	Є проблеми з стабільністю та службою підтримки	+	Є проблеми з роботою чату	+
Популярність на даний час	+	Доволі швидко знижується про що говорять відгуки	Падає і цьому свідчать відгуки користувачів.	+	+
Інші особливості	Сервіс орієнтований на пошук друзів саме за гроші. Про що і говорить назва «Орендуй друга».	Цей сервіс спрямований на організацію зустрічей та подій зі спільними інтересами. Він має широкий спектр тематик та дозволяє створювати свої власні заходи. Крім того, додаток має вбудований календар, що дозволяє зручно планувати зустрічі та події.	Являється моделлю для пошуку друзів, яка входить до додатку Bumble. Загалом ще є Bumble Dating для пошуку відносин та Bumble Bizz для пошуку роботи та працівників	Цей додаток використовує спеціальну систему "Паток-поінтів", які користувач отримує за активну участь у спільноті. Ці "поінти" дозволяють отримувати певні привілеї в додатку та збільшують ймовірність знайти більш сумісних друзів.	Є можливість відправляти подарунки і цікаві листівки своїм друзям та знайомим.

2.АНАЛІЗ ІНСТРУМЕНТІВ РЕАЛІЗАЦІЇ WEB-ДОДАТКУ

Аналіз інструментів для розробки web-додатку є важливим етапом у процесі реалізації web-додатку. Для успішної розробки web-додатку необхідно визначити найбільш підходящі технології та інструменти для використання, враховуючи потреби проекту та вимоги до швидкості та масштабованості.

Він допомагає визначити найкращі практики та підходи для розробки web-додатків відомих на сьогоднішній день

В даній роботі в якості мови програмування було обрано Java.

2.1 Java

Є кілька причин, чому Java є гарним вибором для розробки web-додатку для пошуку друзів та знайомих:

- Сильна спільнота та підтримка - Java має сильну та віддану спільноту розробників, які беруть активну участь у проектах з відкритим вихідним кодом, надають рекомендації та створюють корисні ресурси. Це полегшує пошук допомоги, коли виникають проблеми в процесі розробки.
- Java не залежить від платформи, адже відома своєю філософією "напиши один раз, запускай де завгодно". Це означає, що web-додаток на Java може працювати на будь-якій платформі, яка підтримує віртуальну машину Java (JVM), що включає більшість операційних систем.
- Масштабованість Java добре підходить для створення великомасштабних додатків, які можуть впоратися з високим трафіком і великими робочими навантаженнями. Spring Framework, який зазвичай використовується з Java, надає можливості для створення масштабованих і підтримуваних додатків.
- Java приділяє значну увагу безпеці і включає вбудовані механізми для запобігання поширених вразливостей web-додатків, таких як міжсайтовий скриптинг (XSS) і SQL-ін'єкції.

Щодо каркасу за допомогою якого буде створюватися програмний додаток буде

використано Spring Framework.

Spring Framework надає безліч інструментів і бібліотек, які спрощують і прискорюють процес розробки web-додатків. Деякі з переваг використання Spring Framework включають:

- Інверсія управління та реалізація залежностей - Spring Framework надає механізм інверсії управління, який дозволяє розробникам позбутися жорстких залежностей між компонентами програми. Це полегшує заміну компонентів і спрощує тестування.
- Аспектно-орієнтоване програмування (АОП) - Spring Framework підтримує АОП, що дозволяє розробникам легко реалізовувати функціональність, яка повинна застосовуватися до декількох компонентів програми.
- Підтримка транзакцій - Spring Framework забезпечує підтримку транзакцій, що дозволяє легко керувати транзакціями бази даних.
- Безпека - Spring Framework надає механізми безпеки, такі як автентифікація та авторизація, що дозволяє легко створювати безпечні web-додатки.
- Простота тестування - Spring Framework надає безліч інструментів для тестування web-додатків, що полегшує тестування та зменшує кількість помилок у коді.
- Модульність - Spring Framework складається з декількох модулів, що дозволяє розробникам використовувати лише необхідні компоненти та бібліотеки, що зменшує розмір та складність додатку.

2.2 Архітектура

На рахунок архітектури web-додатку використано один з найпопулярніших і відомих шаблонів.

MVC (Model-View-Controller) - це архітектурний шаблон проектування програмного забезпечення, який широко використовується при розробці web-додатків. Він розділяє додаток на три взаємопов'язані компоненти:

- Модель представляє дані та логіку додатку, які можуть зберігатися в базі даних або іншій системі зберігання даних.
- Представлення репрезентує дані користувачеві та взаємодіє з ними через інтерфейс користувача. Зазвичай реалізується за допомогою HTML, CSS та JavaScript.
- Контролер же обробляє дані, введені користувачем, і вносить зміни в модель і компоненти подання за необхідності. Він є посередником між моделлю та компонентами представлення.

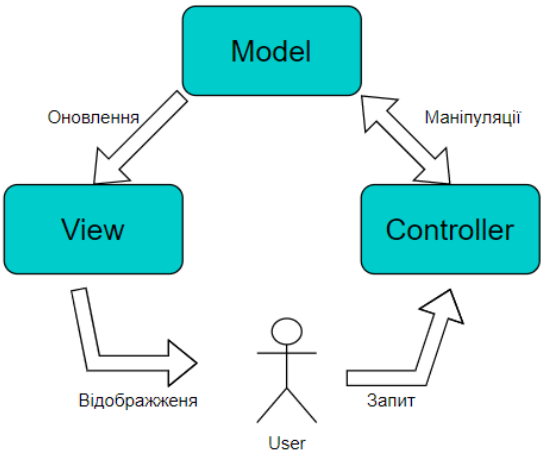


Рисунок 2.1 – Візуальне відображення архітектури MVC

Основна перевага використання шаблону MVC полягає в тому, що він сприяє розділенню завдань, що полегшує підтримку та модифікацію кодової бази. Розробники можуть зосередитися на конкретних компонентах, не турбуючись про функціональність інших компонентів.

2.3 Spring MVC

Компонент контролера реалізується за допомогою web-фреймворку, такого як Spring MVC, який обробляє маршрутизацію запитів і керує потоком даних між моделлю і компонентами представлення.

Spring MVC - це популярний web-фреймворк, побудований на основі Spring

Framework. Він надає реалізацію архітектури Model-View-Controller (MVC) для створення web-додатків на Java. Spring MVC розроблений для спрощення web-розробки і надає кілька функцій, які роблять його популярним вибором серед розробників.

Ось деякі з ключових особливостей Spring MVC:

- Архітектура MVC - Spring MVC слідує архітектурі Model-View-Controller, яка розділяє логіку додатку на три компоненти: модель, представлення та контролер. Таке розділення робить додаток простішим в обслуговуванні та модифікації.
- Ін'єкція залежностей - Spring MVC використовує Dependency Injection (DI) для управління залежностями об'єктів. Це спрощує створення об'єктів, робить додаток більш модульним і легшим для тестування.
- Відображення обробників - Spring MVC використовує механізм відображення обробників для зіставлення вхідних запитів з методами контролерів. Це забезпечує гнучкий та конфігурований спосіб обробки різних типів запитів.
- Роздільна здатність представлення - Spring MVC надає інтерфейс ViewResolver, який дозволяє розробникам вибирати з різних технологій представлення, таких як JSP, Thymeleaf та Mustache.
- Обробка винятків - Spring MVC надає гнучкий та конфігурований механізм обробки винятків, який дозволяє розробникам обробляти винятки у послідовний та структурований спосіб.
- Інтеграція з іншими компонентами Spring Framework - Spring MVC є частиною Spring Framework і легко інтегрується з іншими компонентами Spring, такими як Spring Security та Spring Data.

2.4 Spring Security

В свою чергу для забезпечення безпеки та розмежування доступу до деякого функціоналу web-додатку відповідно до ролей користувачів. Було обрано

фреймворк Spring Security.

Spring Security - це потужний фреймворк безпеки для створення безпечних web-додатків на основі Java. Він є частиною великої екосистеми Spring Framework і надає повний набір функцій та інструментів для захисту web-додатків.

Ось деякі з ключових особливостей Spring Security:

- Аутентифікація та авторизація в Spring Security надає підтримку автентифікації та авторизації для web-додатків. Це дозволяє розробникам аутентифікувати користувачів і контролювати доступ до ресурсів на основі ролей і дозволів користувачів.
- Підтримка аутентифікації із запам'ятовуванням, яка дозволяє користувачам залишатися аутентифікованими навіть після закриття браузера.
- Управління сеансами - Spring Security надає підтримку управління сеансами, що дозволяє розробникам контролювати управління сеансами користувачів, включаючи тайм-аути і анулювання сеансів.
- Захист від підробки міжсайтових запитів (CSRF) - Spring Security надає вбудований захист від CSRF для запобігання атакам, які використовують довіру користувачів до web-додатків.
- Контроль доступу - забезпечує підтримку контролю доступу, що дозволяє розробникам контролювати доступ до ресурсів на основі ролей і дозволів користувачів.
- Інтеграція з іншими компонентами Spring Framework - Spring Security розроблений для безперебійної роботи з іншими компонентами Spring Framework, такими як Spring MVC, Spring Data та Spring Boot.

Підсумовуючи сказане, Spring Security досить багатофункціональний фреймворк. Він надає повний набір функцій та інструментів для захисту web-додатків, включаючи автентифікацію та авторизацію, вхід на основі форм, автентифікацію із запам'ятовуванням, управління сеансами, захист від CSRF, контроль доступу та інтеграцію з іншими компонентами Spring Framework. Spring Security широко використовується у корпоративних web-додатках і підтримується великою та активною спільнотою.

2.5 Hibernate та JPA

У контексті розробки цього додатку компонент моделі буде взаємодіяти з базою даних MySQL за допомогою ORM (об'єктно-реляційне відображення), а саме, Hibernate & JPA. Це дві технології, що використовуються для роботи з базами даних у Java-додатках.

JPA (Java Persistence API) - це стандарт Java EE, який надає специфікацію об'єктно-реляційного відображення (ORM - Object–relational mapping) для відображення об'єктів Java в реляційні бази даних. Він надає високорівневий API, який дозволяє розробникам працювати з базами даних без необхідності писати складні SQL-запити. JPA надає ряд переваг, включаючи покращену організацію коду, скорочення часу розробки та зменшення витрат на обслуговування.

Hibernate - це реалізація JPA з відкритим вихідним кодом, яка надає додаткові можливості та функціональність. Це найпопулярніша реалізація JPA, яка широко використовується в Java-додатках. Hibernate відомий своєю здатністю спрощувати програмування баз даних, надаючи високорівневий API, який абстрагується від основних операторів SQL. Деякі з ключових особливостей Hibernate включають:

- ORM-відображення: Hibernate надає спосіб зіставлення класів Java з таблицями бази даних, що спрощує процес програмування баз даних.
- Кешування: Hibernate надає механізм кешування, який дозволяє розробникам підвищити продуктивність додатків за рахунок зменшення кількості запитів до бази даних.
- Транзакції: Hibernate підтримує транзакції, що забезпечує цілісність даних в базі даних.
- Ліниве завантаження: Hibernate надає механізм лінивого завантаження, який дозволяє більш ефективно використовувати ресурси бази даних, завантажуючи дані тільки тоді, коли вони потрібні.
- Переносимість баз даних: Hibernate забезпечує підтримку широкого спектру баз даних, що дозволяє розробникам переключатися між базами даних без зміни коду програми.

2.6 MySQL

MySQL - це реляційна система управління базами даних (СУБД), яка широко використовується у web-додатках, як внутрішня база даних. Вона була розроблена корпорацією Oracle і випущена в 1995 році.

Вона популярна завдяки своїй масштабованості, продуктивності та простоті використання. Вона добре налаштовується і може бути легко інтегрована з різними мовами програмування та фреймворками, такими як Java, PHP і Python. MySQL також дуже надійна і має перевірений досвід використання у великомасштабних додатках.

Загалом представляє собою клієнт-серверну систему, де сервер бази даних керує зберіганням і пошуком даних, а клієнти взаємодіють з сервером для виконання запитів і маніпулювання даними. MySQL підтримує декілька механізмів зберігання даних, кожен з яких має власний набір функцій і характеристик продуктивності. За замовчуванням використовується механізм зберігання даних InnoDB, який забезпечує підтримку транзакцій і зовнішніх ключів.

MySQL доступна у двох версіях - для спільноти та для підприємств, причому версія для спільноти є безкоштовною у використанні та поширенні. MySQL також надає різноманітні інструменти та утиліти для управління та адміністрування баз даних, включаючи MySQL Workbench, який є візуальним інструментом для проектування, розробки та управління базами даних.

Загалом, MySQL - це популярна і потужна система управління базами даних, яка широко використовується у web-додатках і має багатий набір функцій і можливостей.

2.7 JSP

Щодо компоненту представлення використано механізм шаблонів, JSP, для створення динамічних HTML-сторінок.

JSP (JavaServer Pages) - це технологія, яка дозволяє розробникам створювати динамічні web-сторінки за допомогою Java. JSP схожа на PHP, ASP та інші мови серверних сценаріїв, але використовує синтаксис Java і виконується на стороні сервера. JSP-сторінки компілюються в Java-сервлети і виконуються на сервері, який підтримує Java.

JSP-сторінки складаються з двох частин: статичного HTML-коду та Java-коду, який вбудовується в HTML-код. Код Java виконується на стороні сервера і використовується для генерації динамічного контенту, який відображається на web-сторінці. JSP дозволяє розробникам створювати фрагменти коду багаторазового використання, такі як заголовки, колонтитули та навігаційні панелі, які можна включати в декілька сторінок.

JSP надає кілька переваг, серед яких:

- Інтеграція з Java - JSP-сторінки можуть містити код Java, що полегшує інтеграцію з програмами та фреймворками на основі Java.
- Багаторазове використання - JSP дозволяє розробникам створювати багаторазові фрагменти коду, які можна включати в декілька сторінок, що економить час розробки і зменшує дублювання коду.
- Розділення завдань - JSP відокремлює рівень представлення (HTML) від логіки програми (код Java), що полегшує підтримку та модифікацію коду.
- Легкість у вивченні - JSP базується на синтаксисі HTML і Java, що полегшує його вивчення для розробників, які знайомі з цими технологіями.
- Розширюваність - JSP надає набір стандартних тегів для загальної функціональності, таких як ітерації та умовні оператори, а також дозволяє розробникам створювати власні теги для більш складної функціональності.

Таким чином, JSP - це досить легка в освоєнні технологія, яка дозволяє розробникам створювати динамічні web-сторінки за допомогою Java. Вона надає ряд переваг, серед яких інтеграція з Java, багаторазове використання, розділення

завдань, простота вивчення та розширюваність. JSP широко використовується у корпоративних web-додатках і підтримується багатьма web-серверами та серверами додатків.

2.8 HTML та CSS

HTML (Hypertext Markup Language) і CSS (Cascading Style Sheets) - це дві фундаментальні технології, що використовуються для створення і стилізації web-сторінок. Ось деяка інформація про HTML та CSS:

Можливсості та призначення HTML:

- HTML визначає структуру і зміст web-сторінки. Він складається з ряду елементів, які визначають різні частини web-сторінки, такі як заголовки, абзаци, зображення, посилання, форми тощо.

- Семантична розмітка HTML включає в себе семантичну розмітку, що означає використання тегів, які передають значення вмісту, який вони оточують. Це допомагає пошуковим системам розуміти сторінку і покращує доступність для користувачів з обмеженими можливостями.
- Web-браузери інтерпретують HTML і перетворюють його у візуальне представлення, яке користувачі можуть переглядати і взаємодіяти з ним. HTML - це мова розмітки, а не мова програмування, оскільки вона визначає структуру, а не функціональність web-сторінки.
- HTML розвивався протягом багатьох років, і HTML5 є останньою версією. У HTML5 з'явилися нові елементи, покращилася підтримка мультимедіа, додалася семантика і підвищилася сумісність з браузерами.

Можливості та призначення CSS:

- Презентація та стилізація - CSS відповідає за візуальне представлення

та макет web-сторінки. Він дозволяє розробникам визначати стилі для елементів HTML, таких як шрифти, кольори, поля, відступи, позиціонування тощо.

- Розділення завдань - CSS дозволяє чітко розділити структуру/вміст (HTML) і представлення/стилістику (CSS) web-сторінки. Таке розділення полегшує підтримку та оновлення дизайну web-сайту, не впливаючи на його основну структуру.
- Каскадні таблиці стилів - "Каскадний" аспект CSS відноситься до способу застосування стилів і визначення їх пріоритетів. Кілька правил CSS можуть бути спрямовані на один і той самий елемент, а їх застосування визначається на основі специфіки, успадкування та порядку оголошення.
- Адаптивний дизайн - CSS відіграє вирішальну роль у створенні адаптивного web-дизайну, який адаптується до різних розмірів екранів і пристроїв. Медіа-запити в CSS дозволяють розробникам визначати стилі на основі таких факторів, як ширина, висота та орієнтація екрану.

HTML та CSS працюють разом для створення візуально привабливих і структурованих web-сторінок. HTML визначає зміст і структуру, тоді як CSS керує презентацією та стилем, що дозволяє створювати привабливі та зручні для користувача web-сайти.

3 РОЗРОБКА СТРУКТУРИ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ

Визначення структури web-додатку також відіграє велику роль у забезпеченні його ефективного та гнучкого функціонування. Добре структурований web-додаток дозволяє швидше та ефективніше розробляти, простіше та зрозуміліше керувати ним, а також забезпечує зручну масштабованість та підтримку.

3.1 Основні вимоги до функціоналу

Розробка такого типу web-додатку має на меті полегшити можливість пошуку нових друзів зі спільними інтересами та нових знайомств.

Тому проаналізувавши головні аналоги на сьогоднішній день та інструменти реалізації такого типу web-додатків. Було сформовано такий список головних функціональних та нефункціональних вимог які повинні бути виконані:

- Відображення короткої інформації про зареєстрованих користувачів на головному екрані.
- Можливість реєстрації користувачів в системі
- Можливість авторизації користувачів в системі.
- Доступ до особистого кабінету де користувачі можуть переглянути свою особисту інформацію.
- Можливість редагування особистої інформація для користувача.
- Засоби фільтрації відображення користувачів відповідно вибраних параметрів.

Щодо користувачів які мають роль адміна, то вони матимуть свій функціонал доступний лише для них:

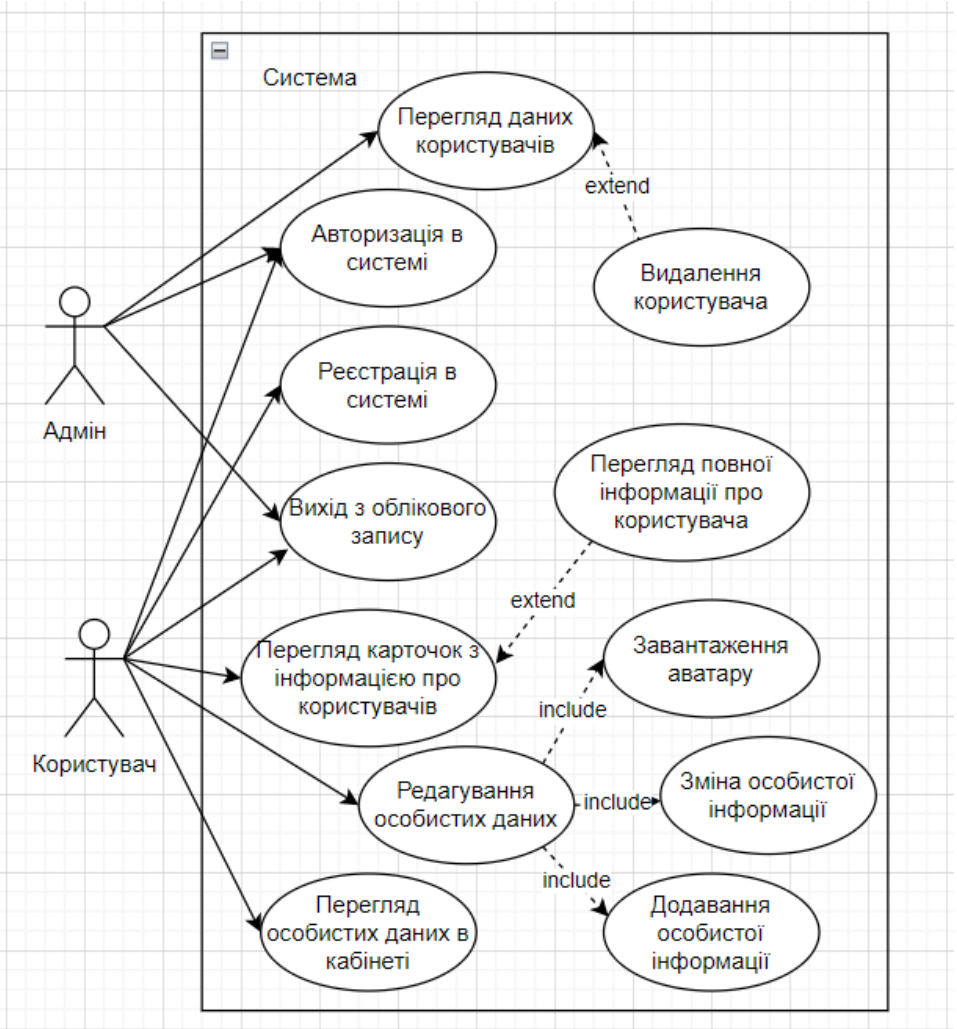
- Окрема сторінка з доступом до даних всіх зареєстрованих користувачів
- Можливість видаляти обліковий засіб користувача.

Нефункціональні вимоги:

- Продуктивність додатку з великою кількістю користувачів і даних.
- Безпека та конфіденційність даних користувачів.
- Сумісність програми з різними браузерами.
- Простота використання та інтуїтивно зрозумілий інтерфейс.
- Масштабованість системи та можливість розвитку додатку в майбутньому.

3.2 Діаграмне представлення

На основі цих вимог також наведено UML діаграму варіантів використання (Use Case Diagram) - це графічний інструмент для моделювання функціональності системи з точки зору її зовнішніх акторів та їх взаємодії з системою. Вона визначає основні функції, які повинна виконувати система, і визначає акторів, які взаємодіють з цими функціями.



3.1 – UML діаграма варіантів використання

А також діаграма діяльності, яка описує послідовність дій, які можна виконати в системі. А також візуалізує бізнес-процеси, алгоритми, логіки роботи системи та взаємодії між об'єктами.

Діаграма діяльності дає можливість описати послідовність дій в процесі, визначити умовні переходи, паралельні виконання та інші аспекти. Вона допомагає зрозуміти логіку процесу, ідентифікувати можливі проблеми або оптимізацій.

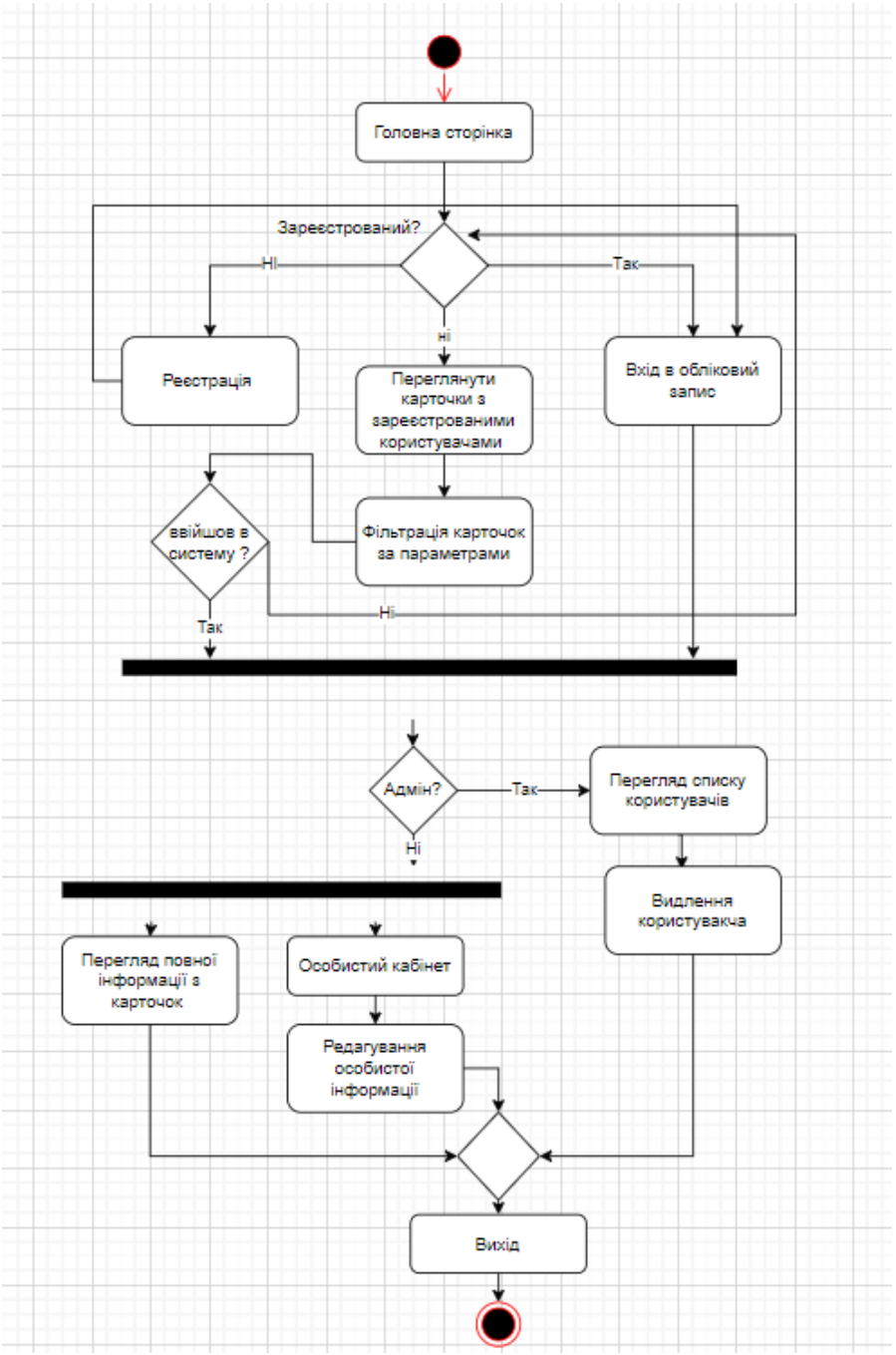


Рисунок 3.2 – Діаграма діяльності

3.3 Представлення внутрішньої структури

Для написання даного web-додатку було використано середовище розробки IntelliJ IDEA від JetBrains. Саме там було створено повну структуру web-додатку. А саме пакети, класи, JSP сторінки та інші ресурси.

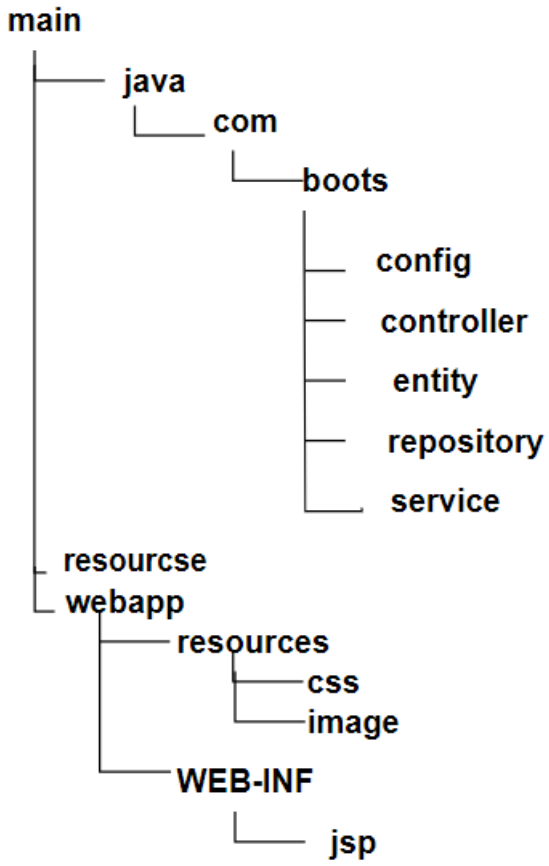


Рисунок 3.3 – структура пакетів проекту

Пакет boots зберігає в собі основні пакети які мають в собі класи та інтерфейси що відповідають за бізнес логіку web-додатку згідно архітектури Spring MVC.

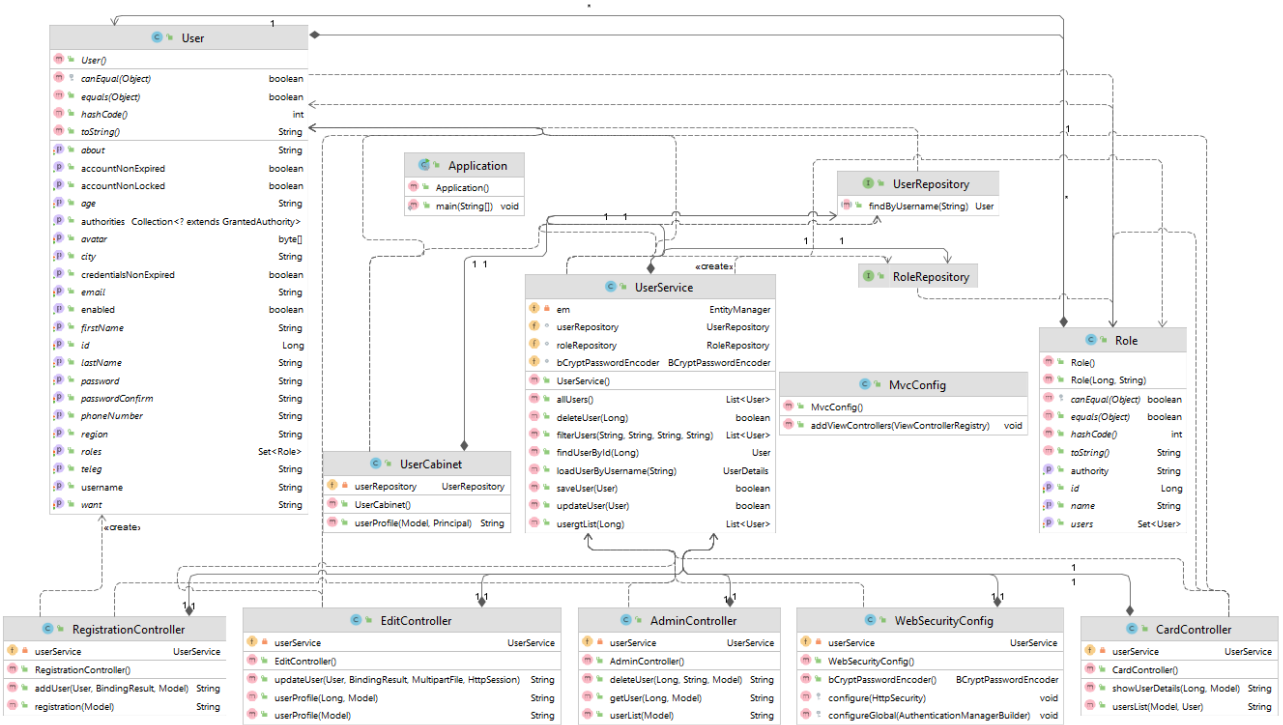


Рисунок 3.4 – Діаграма класів

По порядку що до функціональності яку має додаток і співвідношення класів які реалізують бізнес логіку.

3.3.1 Авторизація та реєстрація

Складові основної логіки, що реалізують реєстрацію та авторизацію користувачів в додатку. Це класи пакету config, контролер що відповідає за логіку реєстрації у пакеті controller та візуальна складова сторінки JSP.

Клас конфігуратор з пакету WebSecurityConfig, який відповідає за налаштування механізму безпеки у web-додатку за допомогою Spring Security. Ось основні аспекти, пов'язані з авторизацією, включені в цей клас.

Основні елементи конфігурації безпеки включають:

- @Configuration і @EnableWebSecurity анотації, які вказують, що клас є класом конфігурації для Spring Security.
- Перевизначення методу configure(HttpSecurity httpSecurity), який визначає правила доступу до ресурсів на основі ролей користувачів.
- Налаштування форми входу (.formLogin()) зі сторінкою входу (/login) і

сторінкою перенаправлення після успішного входу (.defaultSuccessUrl("/")).

- Налаштування механізм виходу із системи (.logout()), включно зі сторінкою виходу із системи та сторінкою перенаправлення після виходу із системи.
- У коді також використовується інший важливий компонент Spring Security, UserDetailsService, який використовує клас userService для отримання інформації про користувача під час аутентифікації.

Також клас конфігуратор McvConfig який відповідає за налаштування шляхів URL і відображень для web-додатку.

Метод addViewControllers викликається для додавання подання для конкретного URL-шляху /login. Зазначене подання з ім'ям "login" буде пов'язане з цим URL-шляхом. За візуальне відображення відповідає сторінка login.jsp пакету webapp.

Рисунок 3.5 – Вигляд сторінки авторизації

Вже за логіку реєстрації відповідає наявний клас контролер з пакету controller під назвою RegistrationController.

Основні методи контролера включають в себе:

- Метод registration() з анотацією @GetMapping("/registration") обробляє GET-запити до шляху /registration. Цей метод створює новий об'єкт User і додає його до моделі, щоб передати його на сторінку registration.jsp для відображення форми реєстрації.
- Метод addUser() з анотацією @PostMapping("/registration") обробляє POST-запити до шляху /registration. Цей метод перевіряє дані реєстраційної форми (userForm), використовуючи анотацію @Valid і BindingResult. Якщо валідація не пройшла успішно, повертається сторінка registration.jsp для відображення помилок. В іншому випадку він перевіряє, чи збігаються паролі, чи існує користувач з таким самим ім'ям, і зберігає нового користувача за допомогою логіки метода saveUser() з класу userService. І в кінці, перенаправляє на домашню сторінку за допомогою redirect:/.

Метод saveUser() в свою чергу зберігає нового користувача в базі даних. Основні кроки методу включають:

- Перевіряється, чи існує користувач із таким самим іменем у базі даних. Якщо такий користувач уже існує, метод повертає false, вказуючи на те, що операція збереження користувача не вдалася.
- Якщо користувача з таким самим ім'ям не існує, то встановлюється роль користувача (наприклад, "ROLE_USER"), пароль шифрується за допомогою bCryptPasswordEncoder і зберігається в базі даних за допомогою userRepository.save(user).
- Нарешті, метод повертає true, щоб показати, що користувача було успішно збережено.
- Цей метод використовує репозиторій userRepository для доступу до бази даних і службу bCryptPasswordEncoder для шифрування пароля користувача перед збереженням.

Рисунок 3.6 – Вигляд сторінки реєстрації.

3.3.2 Картки користувачів та фільтр

Тепер функції фільтрування та відображення карток з короткою інформацією про зареєстрованих користувачів. Все це реалізується за допомогою одного контролера CardController.

Основними методами в контролері є наступні:

- Метод UsersList обробляє GET-запит до шляху "/" (головна сторінка) і відображає карточки користувачів з їх данимим та аватарами. Він також приймає параметри за допомогою filterForm, які використовуються для фільтрації списку користувачів в залежності від обраних параметрів фільтра. Залежно від параметрів фільтра викликається метод filterUsers або allUsers з userService. Результати запиту передаються в модель для подальшого відображення на головній сторінці "index.jsp".

Метод filterUsers у класі UserService відповідає за фільтрацію користувачів залежно від обраних параметрів фільтра.

- Вхідні параметри методу представлені у вигляді рядків age, city, region

і want. Метод використовує ці параметри для фільтрації списку користувачів, який він отримує з userRepository за допомогою методу findAll.

- Кожен параметр фільтра перевіряється на наявність значення, і якщо він не порожній, то фільтр застосовується до списку користувачів відповідно до відповідного критерію. Наприклад, фільтр за віком перевіряє, чи відповідає вік користувача значенню параметра age, і зберігає тільки тих користувачів, чий вік збігається.
- Додаткова фільтрація також виконується для виключення користувачів із роллю "ROLE_ADMIN". Нарешті, метод повертає результуючий список відфільтрованих користувачів.

Метод allUsers у класі UserService повертає список усіх користувачів. Він використовує метод findAll зі сховища userRepository для отримання всіх користувачів з бази даних.

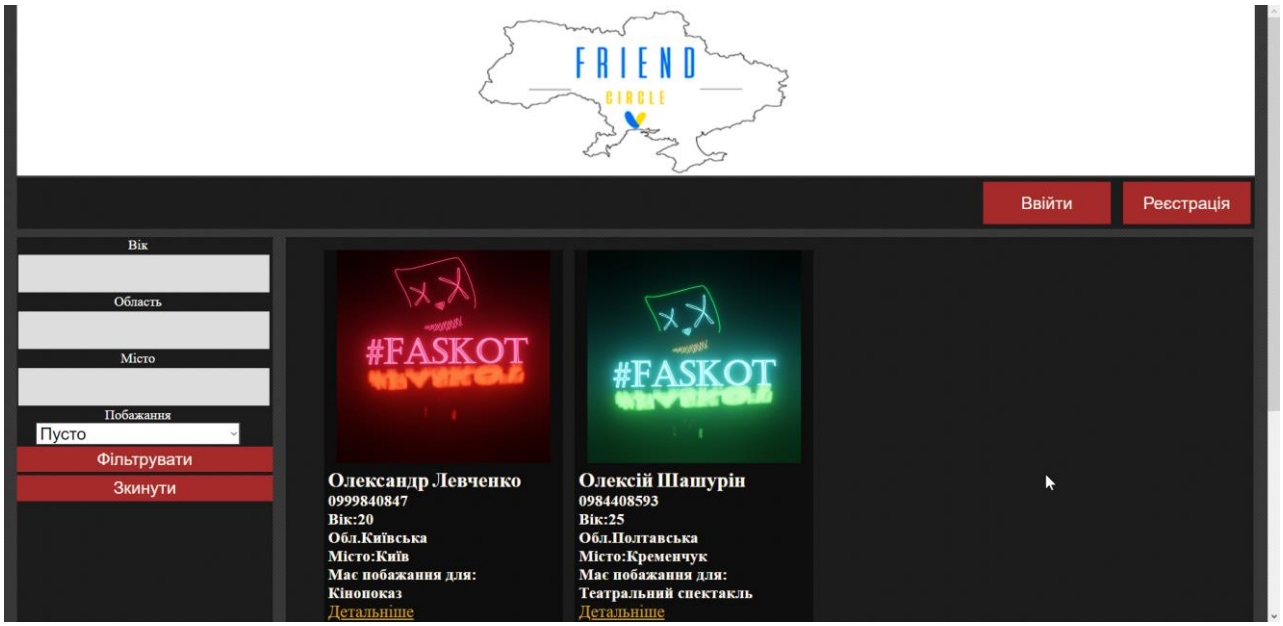


Рисунок 3.7 – Вигляд головного екрану з карточками користувачів та фільтром

3.3.3 Функціонал для авторизованого користувача

Після реєстрації та авторизації користувачу відкривається доступ до іншого функціоналу web-додатку, а саме:

- Перегляд повної інформації про користувача при натисканні кнопки "Детальніше".
- Особистий кабінет в якому він може переглянути інформацію про себе.
- Можливість додавати особисту інформацію про себе та редагувати її.
- Завантаження аватару.

За перегляд повної інформації про користувача, відповідає метод з уже названого контролера CardController.

Його метод showUserDetail обробляє GET-запит за адресою "/about/{id}", де {id} - ідентифікатор користувача. Він відображає деталі про користувача, а також відповідний йому аватар з вказаним ID. Метод викликає findUserById з userService, щоб знайти користувача за його ID. Якщо користувача не знайдено, його буде перенаправлено на сторінку "error.jsp". В іншому випадку дані користувача передаються моделі для відображення на сторінці "about.jsp".

За особистий кабінет користувача відповідає контролер UserCabinet.

У методі userProfile, використовуючи параметр Principal principal, отримують ім'я користувача, яке використовується для пошуку відповідного користувача в базі даних. Використовуючи UserRepository і метод findByUsername, об'єкт користувача витягується з бази даних.

Далі перевіряється, чи є у користувача аватар (поле avatarBytes). Якщо так, то аватар розкодовується з байтового представлення і додається в модель як атрибут "avatarshow" для подальшого відображення на JSP сторінці.

Об'єкт користувача також додається в модель як атрибут "user" що допомагає відобразити інформацію про користувача на тій же JSP сторінці.

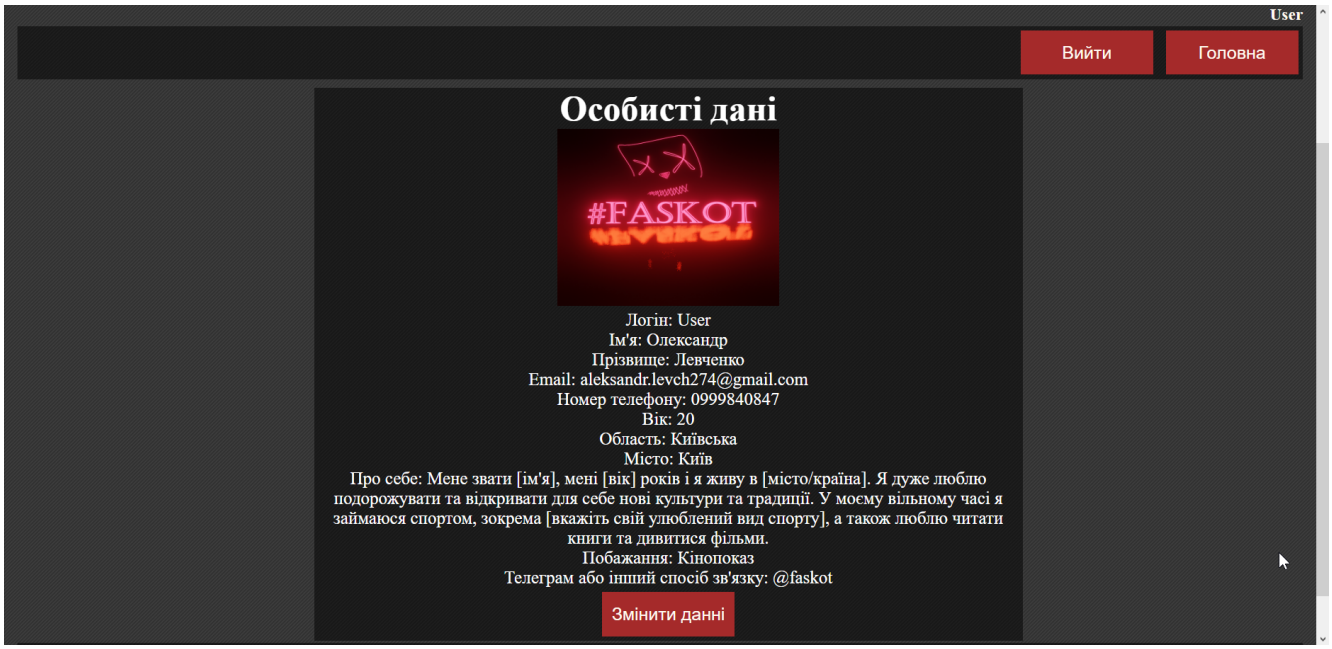


Рисунок 3.8 – Вигляд сторінки особистого кабінету

Тепер в особистому кабінеті доступна можливість змінити дані користуавача, або додати їх, так само завантажити аватар. Цією логікою займається клас котролер EditController.

Метод userProfile з анотацією @GetMapping і параметром @PathVariable("id") Long id отримує ідентифікатор користувача, для якого ви хочете відобразити профіль. Використовуючи userService і метод findUserById, отримайте об'єкт користувача за його ідентифікатором. Цей об'єкт додається до моделі як атрибут "user". Потім метод повертає ім'я відображення "/cabinet", яке відповідає сторінці кабінету користувача для його відображення.

Метод userProfile з анотацією @GetMapping("/editProfile") отримує об'єкт автентифікації користувача з поточного контексту безпеки. З цього об'єкта отримується користувач-принципал, тобто сам об'єкт користувача. Цей об'єкт додається до моделі як атрибут "user". Потім метод повертає ім'я відображення "/editProfile", яке відповідає сторінці редагування профілю користувача.

Метод updateUser з анотацією @PostMapping("/editProfile") оновлює дані профілю користувача. Параметри методу включають об'єкт User для оновлення, об'єкт BindingResult для перевірки помилок валідації, об'єкт MultipartFile для завантаження нового аватара і об'єкт HttpSession для збереження оновленого

об'єкта користувача в сесії.

Якщо в об'єкті result виявлено помилки валідації, метод повертає ім'я відображення "/editProfile" для відображення сторінки редагування з помилками.

Якщо завантажений файл не порожній, його байти отримуються за допомогою file.getBytes() і підставляються до поля аватара об'єкта користувача. В іншому випадку в поле аватара об'єкта користувача встановлюється аватар який вже там є за допомогою existingUser.getAvatar().

Після оновлення профілю за допомогою userService.updateUser(user), оновлений об'єкт користувача зберігається в сеансі за допомогою session.setAttribute("user", user).

На останньому кроці метод повертає перенаправлення на сторінку облікового запису користувача з оновленим ідентифікатором користувача за допомогою user.getId().

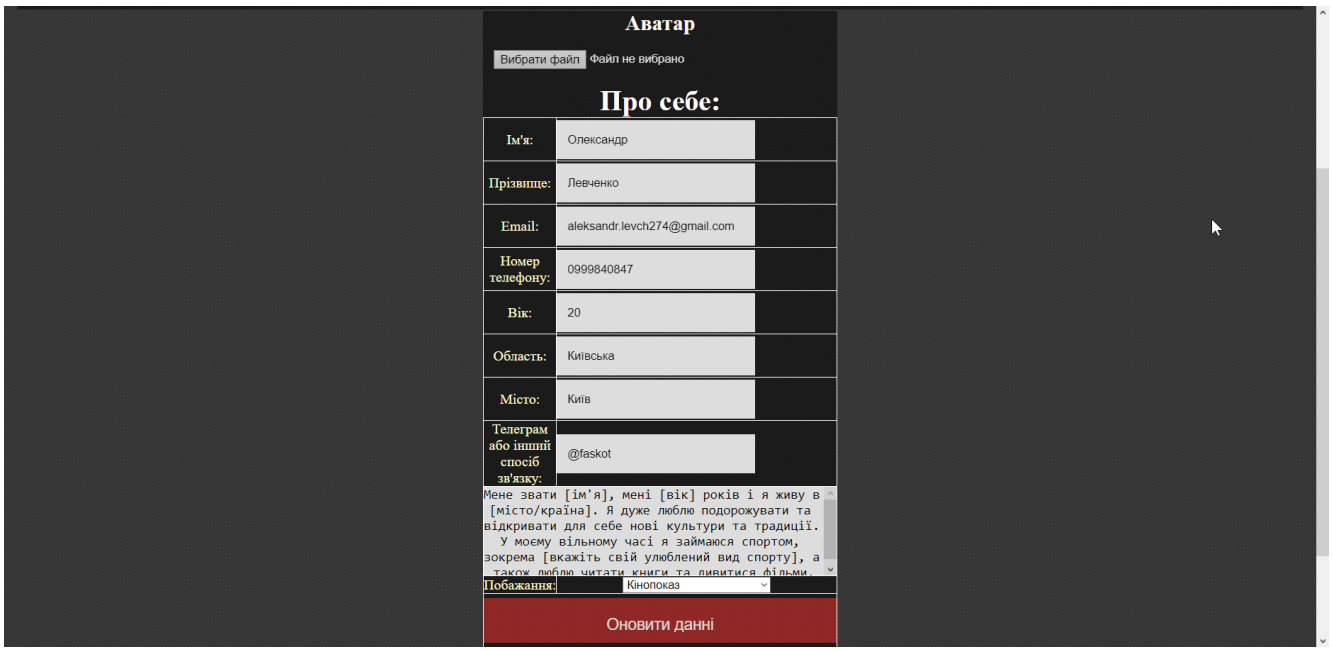


Рисунок 3.9 -Вигляд сторінки для редагування даних

3.3.4 Функціонал для користувача з ролью адмін

В web-додатку згідно діаграми використання ми маємо користувача, який має доступ до спеціального функціоналу. А саме перегляд повного списку користувачів та можливість видалення облікового запису користувача. Весь цей функціонал

реалізує контролер AdminController.

Метод userList з анотацією @GetMapping("/admin") отримує всіх користувачів за допомогою методу allUsers() з userService. Отриманий список користувачів додається до моделі як атрибут "allUsers". Потім метод повертає ім'я відображення "admin", яке відповідає сторінці адміністратора для відображення списку користувачів.

Метод deleteUser з анотацією @PostMapping("/admin") видаляє користувача за його ID. Параметри userId і action методу отримують свої значення з параметрів запиту. Якщо параметром дії є "delete", викликається метод deleteUser(userId) з userService для видалення користувача за його userId. Після видалення користувача метод повертає редирект на сторінку адміністратора "/admin".

Метод getUser з анотацією @GetMapping("/admin/get/{userId}") отримує користувачів з ідентифікатором, більшим за вказане значення userId. Отриманий список користувачів додається до моделі як атрибут "allUsers". Потім метод повертає ім'я відображення "admin", яке відповідає сторінці адміністратора для відображення відфільтрованого списку користувачів.

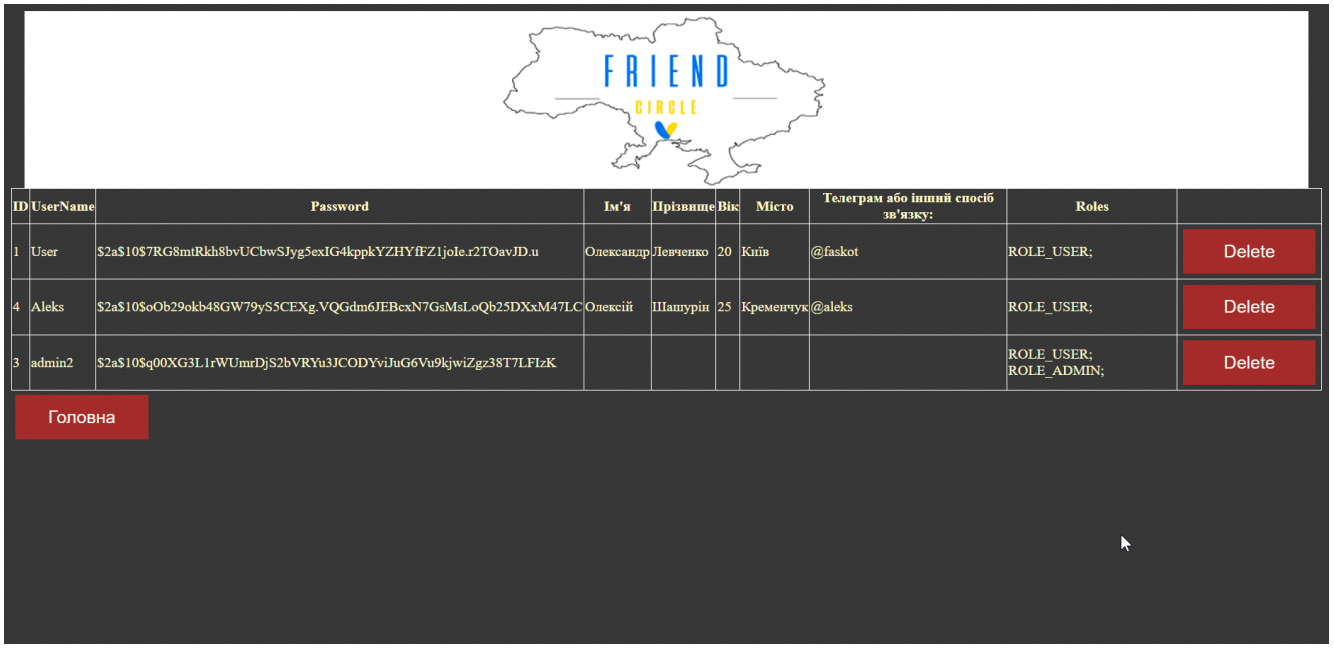


Рисунок 3.10 – Вигляд сторінки адміна

3.4 Представлення структури бази даних.

Щодо зберігання даних web-додатку, та візуальної роботи з базою даних за допомогою графічного інтерфейсу, як уже було описано раніше в аналізі інструментів використано MySQL Workbrench та Hibernate. Для того щоб під'єднати нашу БД до проекту за допомогою Hibernate. Створюються сервер підключений до неї.

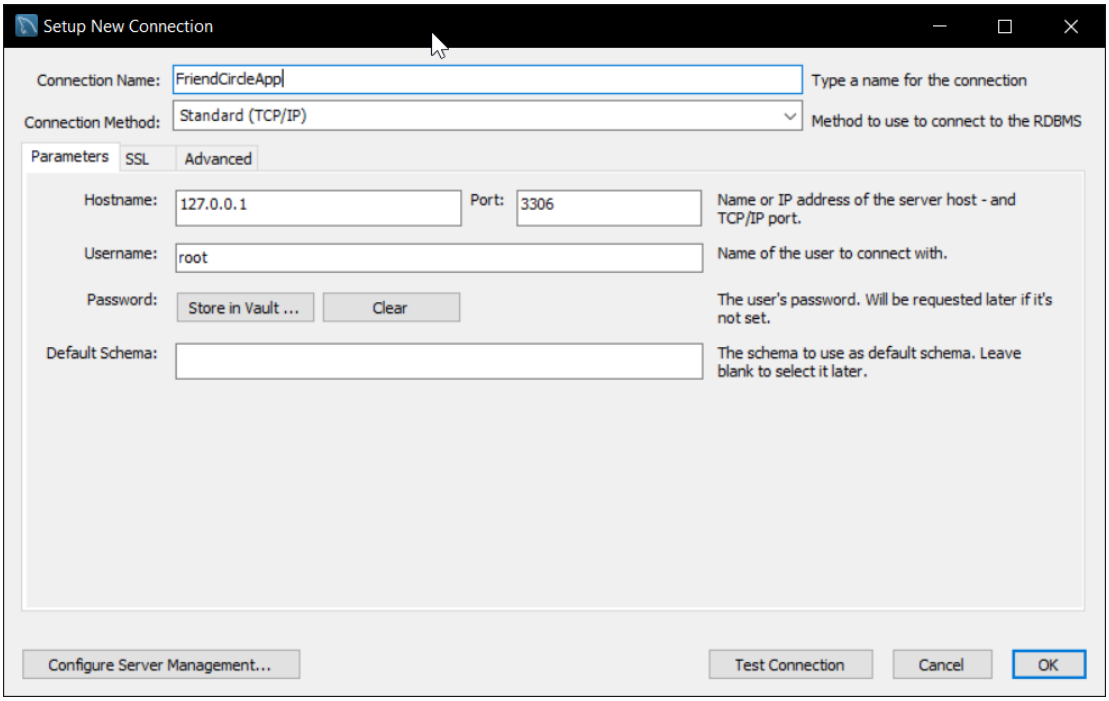


Рисунок 3.11 – створення сервера для БД

Далі просто потрібна пуста базу даних без табличок, бо їх створенням, також взаємодією з ними займається Hibernate у зв'язці з специфікацією JPA.

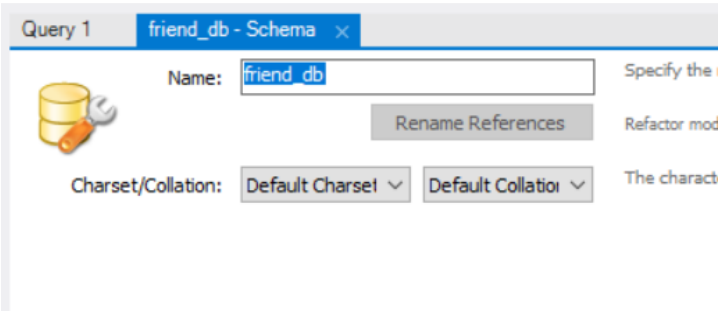


Рисунок 3.12 – створення БД

На цьому можна сказати основна робота з MySQL Workbrench завершена. Далі підключення бази даних до проекту виглядає наступним чином. Прописуємо шлях до нашої бази даних «jdbc:mysql://localhost/friend_db» та авторизуємося на сервері ввівши username та password, root користувача.

application.properties

```
spring.datasource.url=jdbc:mysql://localhost/friend_db
spring.datasource.username=root
spring.datasource.password=mydb_02117678
spring.jpa.show-sql=true
spring.jpa.generate-ddl=false
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
```

Рисунок 3.9 – вигляд коду для підключення та конфігурації БД.

За створення таблиць відповідають Entity на прикладі класу User описано за допомогою чого це відбувається.

Клас User використовує анотації JPA (Java Persistence API) для створення таблиці в базі даних. Основні анотації, що використовуються:

- @Entity: вказує на те, що клас є сутністю, яку потрібно зіставити з таблицею в базі даних.
- @Table: вказує назву таблиці, до якої буде зіставлено клас.
- @Id: вказує поле, яке є первинним ключем у таблиці.
- @GeneratedValue: вказує спосіб генерації значення для первинного ключа.

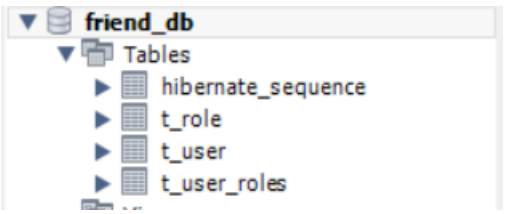
- @Column: вказує додаткові параметри для стовпця в таблиці, такі як назва, тип даних, обмеження тощо.
- @Transient: вказує поле, яке не потрібно зіставляти зі стовпцем у таблиці (не зберігається в базі даних).
- Крім того, анотації JPA використовуються для встановлення зв'язків між таблицями, наприклад, @ManyToMany, @ManyToOne, @OneToMany і @OneToOne, а також анотації для встановлення додаткових параметрів, таких як fetch = FetchType.EAGER або fetch = FetchType.LAZY для визначення стратегії завантаження даних.

Так само створюється таблиця для ролей в якій зберігається найменування ролі та її id. А також таблиця t_user_roles яка є результатом зв'язку "багато до багатьох" між сутностями Користувач і Роль. Коли використовується анотація @ManyToMany між цими двома сутностями, JPA автоматично створює додаткову таблицю для відображення цього зв'язку.

Таблиця t_user_roles має два стовпці:

- user_id - зовнішній ключ, який посилається на поле id в таблиці t_user, що представляє зв'язок з користувачем.
- role_id - зовнішній ключ, який посилається на поле id в таблиці t_role, що представляє зв'язок з роллю.

Ця таблиця забезпечує зв'язок "багато до багатьох" між користувачами та ролями. Кожен запис у таблиці t_user_roles вказує, які ролі призначено конкретному користувачеві. Це досягається за допомогою зовнішніх ключів, які вказують на відповідні записи в таблицях t_user і t_role.



3.12 – Створені таблиці після першого запуску додатку

Тепер щоб призначити роль адміна користувачу потрібно спочатку в таблицю t_role передати id ролей та їх назву.

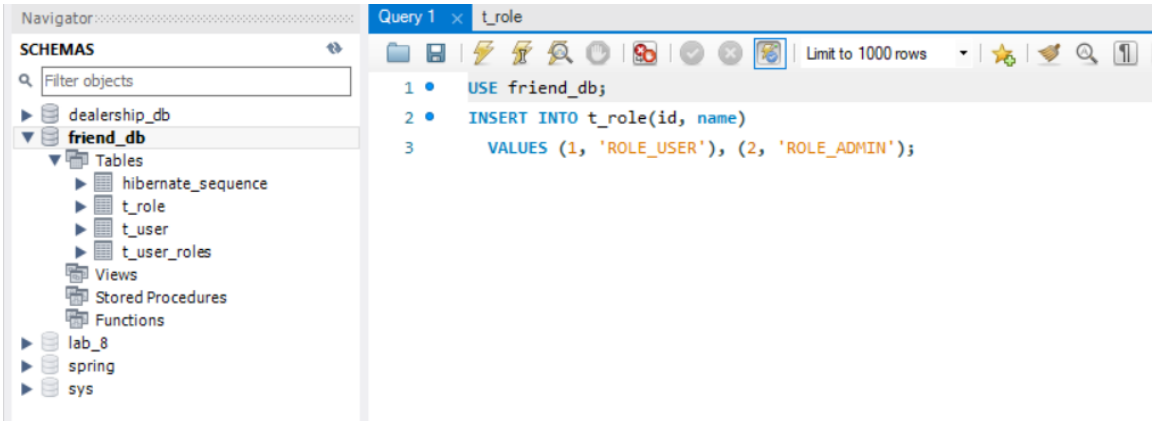


Рисунок 3.13 – Додавання ролей в таблицю t_role

Потім потрібно вибрати користувача якому потрібно видати роль адміна і все готово.

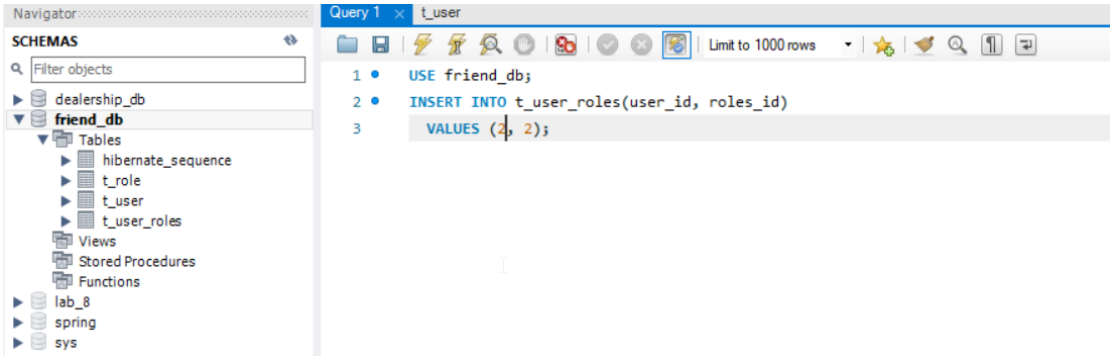


Рисунок 3.14 – Присвоєння ролі адмін користувачеві.

Щодо таблиці hibernate_sequence вона автоматично створюється Hibernate для зберігання інформації про останній згенерований ідентифікатор (ID) для об'єктів, які використовують стратегію генерації значення ID з бази даних.

Коли використовується автоматична генерація ідентифікаторів (наприклад, автоінкремент, послідовність або інший механізм, залежний від бази даних), Hibernate використовує таблицю hibernate_sequence для зберігання останнього використаного значення ідентифікатора. Кожна сутність, яка має автоматично згенерований ідентифікатор, матиме власний рядок у таблиці hibernate_sequence.

Таблиця hibernate_sequence зазвичай має два стовпці:

next_val: значення ідентифікатора, яке буде використано для наступного запису.

назва_послідовності: назва послідовності або генератора, пов'язаного з певною сутністю.

Hibernate використовує цю таблицю для контролю генерації унікальних ідентифікаторів для сутностей з автоматичною генерацією ідентифікаторів.

ВИСНОВКИ

Згідно результатів даної дипломної роботи було розроблено web-додаток для пошуку друзів та знайомств мовою java з використанням Spring framework

В ході виконання дипломної роботи було виконано наступні ключові пункти:

- Проведено аналіз галузі розробки web-додатків для пошуку друзів та знайомств . Її актуальність та ключові проблеми, які можна виділити. Визначено основну ключову аудиторію.
- Розглянуто програмні сервіси аналоги, які можуть бути використано для спрощення пошуку друзів та знайомств. Ключовими недоліками існуючих систем є нав’язування реклами та купівлі платних послуг через обмеження функціональності, також нестабільність роботи деяких версій сервісів на різних платформах. Таким чином, програмний продукт під час розробки покладався на усунення цих недоліків.
- Проаналізовано засоби розробки програмного забезпечення. обрано мову програмування Java, що забезпечує високу надійність та безпеку, особливо це важливо для web-додатків, які містять особисту інформацію користувачів. Крім того, Java пропонує багатий вибір інструментів і фреймворків для розроблення web-додатків, як-от Spring Framework, що забезпечує швидке розроблення та підтримку масштабованих web-додатків. Java також забезпечує легку інтеграцію з іншими технологіями, такими як бази даних і front-end фреймворки.

Створений web-додаток має значний потенціал щодо його покращення в подальшому та додавання іншого функціоналу в залежності від розвитку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Комунікація в Інтернет-просторі: психологічний аспект / Н. І. Лазаренко, А. М. Коломієць, О. М. Паламарчук // Інформаційні технології і засоби навчання. - 2018. - Т. 65, № 3. - С. 249-261. - Режим доступу: http://nbuv.gov.ua/UJRN/ITZN_2018_65_3_20
2. Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools / I.Cosmina, R. Harrop, C. Schaefer, C. Ho., 2017. – 878 с. – (2)
3. Hoffman A. Web Application Security: Exploitation and Countermeasures for Modern Web Applications / Andrew Hoffman., 2020. – 327 с. – (1).
4. Schildt H. Java: A Beginner's Guide / Herbert Schildt., 2018. – 720 с. – (Eighth Edition).
5. Comeau A. MySQL Explained: Your Step By Step Guide to Database Design / Andrew Comeau., 2017. – 392 с.
6. Hergovich P. The Strength of Absent Ties: Social Integration via Online Dating [Електронний ресурс] / P. Hergovich, J. Ortega. – 2018. – Режим доступу до ресурсу: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3044766.
7. Java Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/en/java/.
8. Spring Framework [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/projects/spring-framework.
9. Web MVC framework [Електронний ресурс] – Режим доступу до ресурсу: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html.
10. Allen B. Hibernate Framework in JAVA [Електронний ресурс] / Barry Allen. – 2020. – Режим доступу до ресурсу: https://gocoding.org/hibernate-framework-in-java/.
11. Accessing Data with JPA [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/guides/gs/accessing-data-jpa/.
12. Spring Security [Електронний ресурс] – Режим доступу до ресурсу:

https://docs.spring.io/spring-security/reference/index.html.

13. MySQL Workbench [Електронний ресурс] – Режим доступу до ресурсу: https://dev.mysql.com/doc/workbench/en/.
14. HTML: HyperText Markup Language [Електронний ресурс] – Режим доступу до ресурсу: https://devdocs.io/html/.
15. CSS reference [Електронний ресурс] – Режим доступу до ресурсу: https://devdocs.io/css/.
16. The Java™ Tutorials Lesson: Annotations [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/javase/tutorial/java/annotations/index.html.
17. JavaServer Pages Technology [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/javaee/5/tutorial/doc/bnagx.html.
18. DIGITAL 2022: GLOBAL OVERVIEW REPORT [Електронний ресурс] – Режим доступу до ресурсу: https://datareportal.com/reports/digital-2022-global-overview-report.

ДОДАТОК А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО -НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка web-додатку для пошуку друзів та знайомств мовою Java з використанням Spring framework

Виконав студент 4 курсу

групи ПД-44
Левченко Олександр Олександрович
Керівникроботи

К.т.н, доц, доцент кафедри ІПЗТрінтінаНаталія Альбертівна

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Об’єкт дослідження– процес пошуку нових друзів та знайомств.**
- **Предмет дослідження – спрощення процесу пошуку нових друзів та знайомств, через розробку web-додатку мовою Java з використанням Spring Framework та інших допоміжних інструментів для реалізації цього завдання.**
- **Мета дослідження – допомогти полегшити пошук нових друзів та знайомств для людей за допомогоюweb-додатку.**

ЗАВДАННЯ ДИПЛОМНОЇ РОБОТИ

1. *Проаналізувати актуальність теми та відомі наявні аналоги додатків і сервісів.*
2. *Проаналізувати інструменти реалізації для розробкиweb-додатку.*
3. *Розробити структуру веб-додатку.*
4. *Програмно реалізовати досліджену структуру.*

3

Таблиця порівняння аналогів

Назва	RentAFriend	MeetUp	Bumble BFF	Patook	Znakomka
Платформи	WEB	WEB	Android, IOS, WEB	Android, IOS	WEB
Безкоштовний доступ	+	+	+	+	+
Донат	+	+	+	+	+
Реклама	+	+	-	+	+
Платні підписки	-	+	+(Висока нав'язливість)	Частково	+
Заробіток для юзерів	+	-	-	-	-
Геолокація	-	+	+	+	+
Вбудований чат	-	+	+	+	+

Назва	RentAFriend	MeetUp	Bumble BFF	Patook	Znakomka
Фільтри пошуку	+	+	+	+	+
Створення спільнот	-	+	-	-	-
Стабільність роботи	+	Є проблеми з стабільністю та службою підтримки	+	Є проблеми з роботою чату	+
Популярність на даний час	+	Доволі швидко знижується про що говорять відгуки	Падає і цьому свідчать відгуки користувачів.	+	+

4

Список головних функціональних та нефункціональних вимог

- Відображення короткої інформації про зареєстрованих користувачівна головному екрані.
- Можливістьреєстрації користувачівв системі
- Можливістьавторизації користувачівв системі.
- Доступ до особистого кабінету де користувачіможуть переглянути свою особисту інформацію.
- Можливістьредагуванняособистої інформація для користувача
- Засоби фільтрації відображення користувачіввідповідно вибраних параметрів.

Щодо користувачівякі мають роль адміна, то вони матимутьсвій функціоналдоступний лише для них:

- Окрема сторінка з доступом до даних всіх зареєстрованих користувачів
- Можливістьвидалятиобліковий засіб користувача.

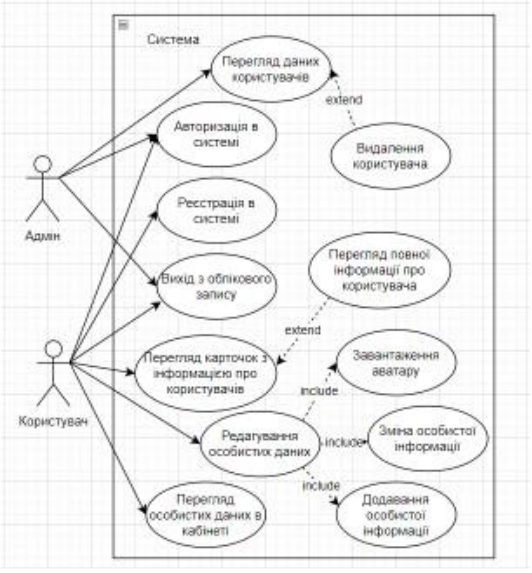
- Нефункціональні вимоги:
- Продуктивністьдодатку з великою кількістю користувачіві даних.
- Безпека та конфіденційність даних користувачів
- Сумісність програмиз різними браузерами
- Простота використання та інтуїтивно зрозумілийінтерфейс.
- Масштабованість системи та можливість розвиткудодатку в майбутньому.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

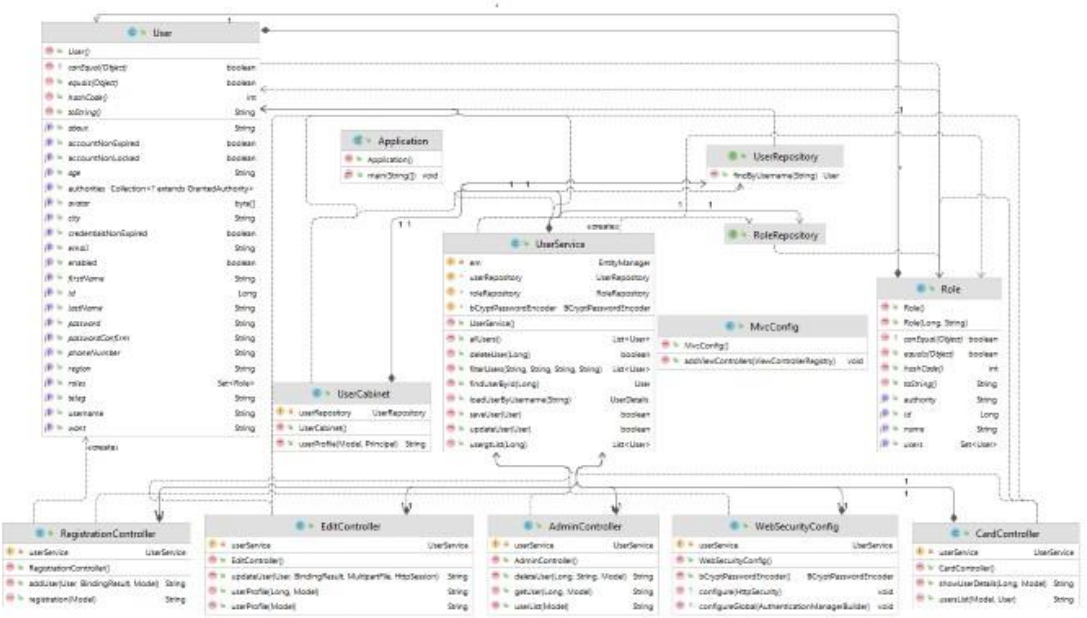
6

Діаграма варіантів використання



7

Діаграма класів



8

ЕКРАННІ ФОРМИ

Головний екран

Сторінка зміни даних профілю

Сторінка реєстрації

Сторінка авторизації

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

- Левченко О.О. Використання Jpa та Hibernate для спрощення роботи з базами даних у веб додатках: Матеріали всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інфокомунікаційних технологіях». Збірник тез - 20.04.2023р, ДУТ, М.Київ.
- Левченко О.О. Безпека та приватність в веб-додатках для знайомств: Матеріали IV всеукраїнської науково-технічна конференція «сучасний стан та перспективи розвитку IOT». Збірник тез - 07.04.2023р, ДУТ, м.Київ. Подано до друку.

ВИСНОВКИ

1. Проведено аналіз галузі розробки web-додатків для пошуку друзів та знайомств. Її актуальність та ключові проблеми, які можна виділити. Визначено основну ключову аудиторію.
2. Розглянуто програмні сервіси аналоги, які можуть бути використано для спрощення пошуку друзів та знайомств.
3. Проаналізовано засоби розробки програмного забезпечення. обрано мову програмування Java, що забезпечує високу надійність та безпеку.
4. Створений web-додаток має значний потенціал щодо його покращення в подальшому та додавання іншого функціоналу в залежності від розвитку.

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б

Проект на git - https://github.com/Faskot/FriendCircle.git

Основні класи

```
@Controller
public class AdminController {
    @Autowired
    private UserService userService;

    @GetMapping("/admin")
    public String userList(Model model) {
        model.addAttribute("allUsers", userService.allUsers());
        return "admin";
    }

    @PostMapping("/admin")
    public String  deleteUser(@RequestParam(required = true, defaultValue = "" )
Long userId,
                              @RequestParam(required = true, defaultValue = "" )
String action,
                              Model model) {
        if (action.equals("delete")){
            userService.deleteUser(userId);
        }
        return "redirect:/admin";
    }

    @GetMapping("/admin/get/{userId}")
    public String  getUser(@PathVariable("userId") Long userId, Model model) {
        model.addAttribute("allUsers", userService.usergtList(userId));
        return "admin";
    }

}


@Controller
public class CardController {

    @Autowired
    private UserService userService;

    @GetMapping("/")
    public String usersList(Model model, @ModelAttribute("filterForm") User
filterForm ) {
        List<User> users;
        if (filterForm.getAge() != null || filterForm.getCity() != null ||
filterForm.getWant() != null) {
            users = userService.filterUsers(filterForm.getAge(),
filterForm.getCity(), filterForm.getRegion(), filterForm.getWant());

        } else {
            users = userService.allUsers().stream()
                    .filter(u -> u.getRoles().stream()
                            .noneMatch(role ->
role.getName().equals("ROLE_ADMIN")))
                    .collect(Collectors.toList());
        }
        // console check
```

```
        System.out.println("Age: " + filterForm.getAge());
        System.out.println("City: " + filterForm.getCity());
        System.out.println("Region" + filterForm.getRegion());
        System.out.println("Want: " + filterForm.getWant());


        Map<Long, String> avatars = new HashMap<>();
        for (User user : users) {
            if (user.getAvatar() != null) {
                String base64Avatar =
Base64.getEncoder().encodeToString(user.getAvatar());
                avatars.put(user.getId(), base64Avatar);
            }
        }

        model.addAttribute("avatars", avatars);
        model.addAttribute("allUsers", users);

        return "index";
    }
    @GetMapping("/about/{id}")
    public String showUserDetails(@PathVariable("id") Long userId, Model model) {
        User userAbout = userService.findUserById(userId);
        if (userAbout == null) {
            return "error";
        }
        String base64Avatar =
Base64.getEncoder().encodeToString(userAbout.getAvatar());

        model.addAttribute("avatarshow", base64Avatar);
        model.addAttribute("userAbout", userAbout);
        return "about";
    }
}



@Controller
public class EditController {
    @Autowired
    private UserService userService;

    @GetMapping("/cabinet/{id}")
    public String userProfile(@PathVariable("id") Long id, Model model) {
        User user = userService.findUserById(id);
        model.addAttribute("user", user);
        return "/cabinet";
    }

    @GetMapping("/editProfile")
    public String userProfile(Model model) {
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        User user = (User) authentication.getPrincipal();
        model.addAttribute("user", user);
        return "/editProfile";
    }

    @PostMapping("/editProfile")
    public String updateUser(@ModelAttribute("user") User user, BindingResult
result, @RequestParam("file") MultipartFile file, HttpSession session) throws
IOException {
        User existingUser = userService.findUserById(user.getId());
        if (result.hasErrors()) {
```

```
            return "/editProfile";
        }
        if (!file.isEmpty()) {
            byte[] bytes = file.getBytes();
            user.setAvatar(bytes);
        }
        else {user.setAvatar(existingUser.getAvatar());}

        userService.updateUser(user);
        session.setAttribute("user", user);
        return "redirect:/cabinet/" + user.getId();
    }

}


@Controller
public class RegistrationController {

    @Autowired
    private UserService userService;


    @GetMapping("/registration")
    public String registration(Model model) {
        model.addAttribute("userForm", new User());
        return "registration";
    }

    @PostMapping("/registration")
    public String addUser(@ModelAttribute("userForm") @Valid User userForm,
BindingResult bindingResult, Model model) {

        if (bindingResult.hasErrors()) {
            return "registration";
        }
        if (!userForm.getPassword().equals(userForm.getPasswordConfirm())){
            model.addAttribute("passwordError", "Паролі не співпадають");
            return "registration";
        }
        if (!userService.saveUser(userForm)){
            model.addAttribute("usernameError", "Користувач з таким іменем вже
існує");
            return "registration";
        }

        return "redirect:/";
    }
}


@Controller
public class UserCabinet {

      @Autowired
    private UserRepository userRepository;

    @GetMapping("/cabinet")
    public String userProfile(Model model, Principal principal) {
        String username = principal.getName();
        User user = userRepository.findByUsername(username);
        byte[] avatarBytes = user.getAvatar();
        if (avatarBytes != null) {
```

```
            String base64Avatar = Base64.getEncoder().encodeToString(avatarBytes);
            model.addAttribute("avatarshow", base64Avatar);
        }
        model.addAttribute("user", user);
        return "cabinet";
    }
}


@Service
public class UserService implements UserDetailsService {
    @PersistenceContext
    private EntityManager em;
    @Autowired
    UserRepository userRepository;
    @Autowired
    RoleRepository roleRepository;
    @Autowired
    BCryptPasswordEncoder bCryptPasswordEncoder;


    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        User user = userRepository.findByUsername(username);
        if (user == null) {
            throw new UsernameNotFoundException("User not found");
        }
        return user;
    }
    public User findUserById(Long id) {
        return userRepository.findById(id).orElse(null);
    }

    public boolean updateUser(User updatedUser) {
        User userFromDB =
userRepository.findById(updatedUser.getId()).orElse(null);
        if (userFromDB != null) {
            userFromDB.setFirstName(updatedUser.getFirstName());
            userFromDB.setLastName(updatedUser.getLastName());
            userFromDB.setPhoneNumber(updatedUser.getPhoneNumber());
            userFromDB.setEmail(updatedUser.getEmail());
            userFromDB.setAvatar(updatedUser.getAvatar());
            userFromDB.setAge(updatedUser.getAge());
            userFromDB.setCity(updatedUser.getCity());
            userFromDB.setRegion(updatedUser.getRegion());
            userFromDB.setAbout(updatedUser.getAbout());
            userFromDB.setWant(updatedUser.getWant());
            userFromDB.setTeleg(updatedUser.getTeleg());
            userRepository.save(userFromDB);
            return true;
        }
        return false;
    }

    public List<User> filterUsers(String age, String city, String region, String
want) {
        List<User> users = userRepository.findAll();

        if (age != null && !age.isEmpty()) {
            users = users.stream()
                    .filter(user -> user.getAge() != null &&
user.getAge().equals(age))
```

```
                    .collect(Collectors.toList());
        }

        if (city != null && !city.isEmpty()) {
            users = users.stream()
                    .filter(user -> user.getCity() != null &&
user.getCity().equalsIgnoreCase(city))
                    .collect(Collectors.toList());
        }

        if (region != null && !region.isEmpty()) {
            users = users.stream()
                    .filter(user -> user.getRegion() != null &&
user.getRegion().equalsIgnoreCase(region))
                    .collect(Collectors.toList());
        }

        if (want != null && !want.isEmpty()) {
            users = users.stream()
                    .filter(user -> user.getWant() != null &&
user.getWant().equalsIgnoreCase(want))
                    .collect(Collectors.toList());
        }
        users = users.stream()
                .filter(user -> user.getRoles().stream().noneMatch(role ->
role.getName().equals("ROLE_ADMIN")))
                .collect(Collectors.toList());
        return users;
    }


    public List<User> allUsers() {
        return userRepository.findAll();
    }

    public boolean saveUser(User user) {
        User userFromDB = userRepository.findByUsername(user.getUsername());

        if (userFromDB != null) {
            return false;
        }

        user.setRoles(Collections.singleton(new Role(1L, "ROLE_USER")));
        user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));
        userRepository.save(user);
        return true;
    }


    public boolean deleteUser(Long userId) {
        if (userRepository.findById(userId).isPresent()) {
            userRepository.deleteById(userId);
            return true;
        }
        return false;
    }

    public List<User> usergtList(Long idMin) {
        return em.createQuery("SELECT u FROM User u WHERE u.id > :paramId",
User.class)
                .setParameter("paramId", idMin).getResultList();
    }
}
```